PingFederate Server

June 25, 2025

PINGFEDERATE SERVER

Version: 11.3

Copyright

Refer to https://docs.pingidentity.com for the most current product documentation.

Trademark

Disclaimer

The information provided in Ping Identity product documentation is provided "as is" without warranty of any kind. Ping Identity disclaims all warranties, either express or implied, including the warranties of merchantability and fitness for a particular purpose. In no event shall Ping Identity or its suppliers be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages, even if Ping Identity or its suppliers have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of liability for consequential or incidental damages so the foregoing limitation may not apply.

Table of Contents

PingFederate

PingFederate is an enterprise federation server that enables user authentication and single sign-on (SSO). It serves as a global authentication authority that lets your customers, employees, and partners securely access applications.

Release Notes

- Current
- Previous releases

Get Started with PingFederate

- Introduction to PingFederate
- Installing and uninstalling PingFederate
- PingFederate administrative console

Use PingFederate

- Administrator's Reference Guide
- SDK Developer's Guide
- Developer's Reference Guide
- Integrations

Troubleshoot PingFederate

- Enabling debug messages and console logging
- Resolving startup issues
- Troubleshooting data store issues
- Resolving URL-related errors
- Resolving service-related errors
- Troubleshooting authentication policy issues
- Troubleshooting registration and profile management issues
- Troubleshooting runtime errors
- Troubleshooting OAuth transactions
- Other runtime issues
- Collecting support data

Learn More

- Community
- Support
- PingFederate customer training (existing customers only)
- Partner Portal (partners)

Release Notes

These release notes summarize the changes in current and previous product updates.

PingFederate enables outbound and inbound solutions for single sign-on (SSO), federated identity management, mobile identity security, API security, social identity integration, and customer identity and access management (CIAM). PingFederate extends employee, customer, and partner identities across domains without passwords, using only standard identity protocols: SAML, WS-Federation, WS-Trust, OAuth, and System for Cross-domain Identity Management (SCIM).

PingFederate 11.3.12 (May 2025)

Resolved issues

Unnecessary ID token reissued with secondary client secret

Fixed PF-37450

We've fixed a defect that caused the token endpoint to unnecessarily reissue an ID token when using a secondary client secret and an asymmetric algorithm for token signing and encryption.

PingFederate 11.3.11 (April 2025)

New features and enhancements

NATIVE_S3_PING update

Improved PF-37234

We've updated the behavior of the `NATIVE_S3_PING` discovery protocol when the `remove_all_data_on_view_change` parameter is active.

Previously, the protocol would delete all files in the S3 bucket, which could lead to the creation of an unwanted subcluster.

Now the protocol deletes all files except for its own to prevent the S3 bucket from being empty.

Learn more in Dynamic cluster discovery.

Resolved issues

Group membership loss during provisioning

Fixed PF-36874

We've fixed a defect that caused PingFederate to lose user group membership information when it lost contact with the datastore during provisioning operations.

Secondary secret missing ID token claim

Fixed PF-37279

We've fixed a defect that caused the ID token claim to be omitted when an OAuth client uses the secondary secret.

PingFederate 11.3.10 (December 2024)

Resolved issues

Cross-site scripting

Security PF-36304 PF-36311 PF-36313

We've fixed a security vulnerability where PingFederate accepted cross-site scripting inputs.

Email verification failure after registration workflow

Fixed PF-36574

We've fixed a defect that caused the email verification screen to fail to appear when a user registered through an authentication source.

OAuth Client Set Authentication Selector with DynamoDB

Fixed PF-36662

We've fixed a defect that caused an error in searching for OAuth Client for OAuth Client Set Authentication Selector when DynamoDB is the client storage.

PingFederate 11.3.9 (November 2024)

Resolved issues

Refresh token time zone discrepancies

Fixed PF-35867

We've fixed a defect that caused refresh tokens to roll prematurely when making authorization requests to servers in different time zones.

Provisioning character limit

Fixed PF-36035

We've fixed a defect that caused outbound provisioning to fail and cease if a source user object exceeded a 255-character limit. In the new behavior, PingFederate will skip user objects that exceed 255 characters and log a warning.

PingDirectory password warning

Fixed PF-36232

We've fixed a defect that prevented PingFederate from issuing a password expiration warning when using PingDirectory as a datastore.

Multiple application requests within a browser

Fixed PF-36239

We've fixed a defect that could cause inconsistent sessions or authentication errors when starting multiple applications in different browser tabs at the same time.

Device authorization grant time zone error

Fixed PF-36261

We've fixed a defect that caused device authorization grant flow errors when clustered server nodes are in different time zones.

PingFederate 11.3.8 (July 2024)

Resolved issues

OIDC admin login failure

Fixed PF-34523

We've fixed a defect that caused the OIDC administrative console login to fail when the `node.group.id` didn't match a server's node id.

OGNL `Extended Property` retrieval failure

Fixed PF-35111

We've fixed a defect that caused OGNL to fail to obtain the `Extended Property` value in authorization policies or fragments.

Refresh token rolls when configured not to roll

Fixed PF-35166

We've fixed a defect that caused PingFederate to roll refresh tokens when **Refresh Token Rolling Policy** is disabled but **Refresh Token Rolling Interval** has a value.

OAuth client only validates one access token manager when `aud` parameter included

Fixed PF-35737

We've fixed a defect that caused PingFederate to validate only the first OAuth client access token manager it found when **Validate Against All Eligible Access Token Managers** was checked, and the `aud` parameter was included in the request.

Custom adapter not returning IPv4 addresses

Fixed PF-35783

We've fixed a defect where PingFederate failed to return IPv4 addresses in a custom adapter request using the `request.getRemoteAddr()` method.

Davinci integration kit

Info PF-35838

The Davinci integration kit has been updated to version 1.2.

PingFederate 11.3.7 (May 2024)

New features and enhancements

PingOne admin URL property

New PF-31859

Added support for the Australia region to the `pf.pingone.admin.url.region` property.

The Asia region is deprecated. We recommend using the Australia region instead.

To learn more, see Configuring PingFederate properties.

Resolved issues

Authentication API allows different user for change password flow

Fixed PF-35609

Fixed a defect that caused the authentication API to allow a different user to proceed with the `MUST_CHANGE_PASSWORD` function than the user who initiated the flow.

Note that in all cases, the target user's password was required to complete the change password operation.

Memory heap increase when using admin API on policy tree

Fixed PF-35423

Fixed a defect that caused PingFederate not to release memory when using the admin API on the policy tree.

Authentication API password change flow ignores credentials

Fixed PF-35618

Fixed a defect that caused the authentication API to ignore credentials for password changes provided after user authentication.

Authentication API validation error

Fixed PF-35430

Fixed a defect that caused a validation error in the authentication API when including the `ui_locales` parameter.

Provisioner uses wrong time zone when datasource and PingFederate are in different time zones

Fixed PF-35286

Fixed a defect that caused redundant user provisioner updates when the datasource and PingFederate were in different time zones.

PingFederate 11.3.6 (April 2024)

Resolved issues

Java thread exhaustion in PingOne Advanced Services

Fixed PF-35411

Fixed a defect that caused repeated looping in authentication policy involving a local Identity profile.

OAuth clients in use detection

Fixed PF-35407

Fixed a defect with In Use detection when DynamoDB is used for OAuth client storage.

OIDC policy DELETE request timeout

Fixed PF-35357

Fixed a defect where deleting an OIDC Policy fails when using DynamoDB storage for a large number of OAuth clients.

Active Directory binary attribute caused thread proliferation

Fixed PF-35142

Fixed a defect that caused LDAP data source connection pools to close when still in use after the LDAP data source is modified and replicating under heavy load.

JWKS algorithm parameter not populated after processing shared keys from cluster

Fixed PF-35309

Fixed a defect that caused the `alg` parameter to fail to populate when EC dynamic keys are rotated on a lead cluster node and shared to the cluster.

Upgraded Jetty Library

Improved PF-35184

Upgraded the Jetty library to version 9.4.54.v20240208.

Lightning LDAP library

Improved PF-35310

Upgraded the lightning LDAP library to version 1.5.22.

PingFederate 11.3.5 (February 2024)

Resolved issues

Rest datastore security vulnerability

Security PF-34720

Fixed a JSON injection vulnerability in REST datastores described in security advisory SECADV044.

Runtime nodes security vulnerability

Security PF-34896

Fixed a path traversal vulnerability in Runtime nodes described in security advisory SECADV044.

OpenID Connect policy management editor security vulnerability

Security PF-35081

Fixed a Cross-Site Scripting vulnerability in the OpenID Connect Policy Management Editor described in security advisory SECADV044.

GET SAML request signature processing error

Fixed PF-34641

Fixed a defect where SAML request using HTTP GET method with multiple signature-related parameters encoded in the *RelayState* parameter were causing errors in processing signature validation.

NPE notification error

Fixed PF-34813

Fixed a defect that caused PingFederate to issue null pointer exception (NPE) errors when querying the token endpoint.

Reencyption causes connection or client to fail on engine

Fixed PF-34409

Fixed a defect where changes made on the administrative console were not replicated to the engine during reencryption.

JMX registration failure for imported archives

Fixed PF-34796

Fixed a defect that caused the JMX monitoring to fail to register archive files that are imported to PingFederate.

Content type changes if `well_known` endpoint response is too large

Fixed PF-34865

Fixed a defect that caused the `content-type` of a response from the `well_known` endpoint to change from JSON to HTML if a response is too large.

RHEL 8 using OS-level FIPS causes PingFederate failure

Fixed PF-34879

Fixed a defect that caused PingFederate to fail on startup when installed on a Red Hat Enterprise Linux (RHEL) server with OS-levels FIPS enabled.

Unable to deobfuscate grant attributes

Fixed PF-34839

Fixed a defect where PingFederate was unable to deobfuscate grant attributes of a certain length.

Valid Authorization policy generates "Configuration Error" message

Fixed PF-34853

Fixed a defect that caused PingFederate to incorrectly return an "Invalid Configuration" error for a valid authentication policy.

PingFederate 11.3.4 (December 2023)

Resolved issues

Fixed JDK8 cluster node issue

Fixed PF-34837

Fixed an issue where nodes were not able to join a cluster when running with JDK8.

PingFederate 11.3.3 (November 2023)

Resolved issues

Improved client authentication security

Security PF-34645

Fixed a potential security vulnerability described in security advisory SECADV040.

Added support for partitioned cookies

New PF-34440

PingFederate now supports using the `Partitioned` attribute to address third-party cookie issues with the iframe-based login widgets in Google Chrome.

Fixed `/idp/startSLO.ping` `404` caused by virtual issuer configuration

Fixed PF-34322

Fixed an issue that was returning a `404` error if the `/idp/startSLO.ping` endpoint was hit while a virtual issuer was configured. You can now configure virtual issuers with a context path.

Client JWKS now sets properly when using DynamoDB storage

Fixed PF-34504

Clients that maintain a JWKS endpoint can now use private key JWT based authentication when requesting an access token.

Fixed NPE when checking an existing persistent grant that is expired with DynamoDB

Fixed PF-34606

Checking for existing but expired grants with DynamoDB no longer causes a null pointer exception error (NPE).

Connections close after getting a `401` or `403` from PingOne API

Fixed PF-34545

Fixed an issue preventing PingFederate from closing connections after receiving a `401` or `403` response from PingOne MFA.

Outbound provisioning performance improvement

Fixed PF-33466

You can now turn off server-side sorting for LDAP requests related to outbound provisioning, which can improve performance in some environments.

Configure this option using the `ProvisionWithServerSort` parameter in the `com.pingidentity.common.util.ldap.LDAPUtil.xml` file.

Unable to copy and paste policy contract in specific situations

Fixed PF-34433

You can now copy and paste a policy contract below a selector node.

XML decryption failing with `KeyName` element

Fixed PF-34536

Fixed an issue where decryption of an encrypted SAML element could fail if a `KeyName` was specified.

Resolved a vulnerability in the Initial Setup Wizard

Security PF-34646

Fixed a Server-Side Request Forgery vulnerability in the Initial Setup Wizard described in security advisory SECADV041.

Certificate in-use detection slowdown

Fixed PF-34077

We fixed a defect that caused PingFederate to check every certificate when loading certificate-related pages in the administrative interface, which slowed down performance.

Upgraded third-party libraries

Improved

- Upgraded Jetty to version 9.4.53.v20231009.
- Upgraded JGroups to version 4.2.24.Final.

PingFederate 11.3.2 (September 2023)

New features and enhancements

Authenticating to Azure SQL Managed Instance through Azure Active Directory

Improved

Now PingFederate supports authentication to Azure SQL Managed Instance through Azure Active Directory without a username and password. For more information, see Configuring a JDBC connection.

Jetty library upgrade

Improved

We upgraded the Jetty library to 9.4.52.v20230823.

Resolved issues

One-time link in password-reset email messages

Fixed PF-33983

When using redirectless mode, now the one-time link (OTL) in password-reset email messages returns users to the authentication API application configured for the policy, rather than to PingFederate.

Incorrect error template when using service provider authentication policies

Fixed PF-34111

When a service provider (SP) authentication policy fails, PingFederate now renders the `sp.sso.error.page.template.html` page instead of the `idp.sso.error.page.template.html` page.

Updating OAuth clients with dynamic client registration

Fixed PF-34146

Fixed a defect where an OAuth client created with dynamic client registration (DCR) couldn't be updated with DCR after it was modified with the administrative console.

Idle JDBC datastore connections

Fixed PF-34163

Now PingFederate closes idle JDBC datastore connections until the minimum pool size is reached instead of closing and recreating all of them.

The `id_token_jti` property in token endpoint responses

Fixed PF-34210

The `id_token_jti` property is no longer included in token endpoint responses.

Administrative API defect when fragment rules have Default to Success disabled

Fixed PF-34216

Fixed an administrative API defect when a fragment rule had **Default to Success** disabled

Email notifications for licensing events even when disabled

Fixed PF-34225

Resolved an issue that caused PingFederate to send email notifications for licensing events even though they were disabled in the **Runtime Notifications** configuration.

PingFederate 11.3.1 (August 2023)

New features and enhancements

Configuration retrieval on engine start up

Improved PF-33667

We introduced new settings in the `cluster-config-replication.conf` file to improve configuration retrieval reliability during engine startup. By setting *publish.replication.data.on.startup* to `true`, the administrative console automatically publishes the last replicated configuration upon startup, eliminating the need to initiate replication through the administrative UI or API after a console restart. Additionally, you can configure engines to fail startup if they cannot retrieve configuration data by setting *require.replication.data.on.startup* to `true`. This setting proves beneficial in DevOps deployments, where fresh engine nodes are frequently created without any initial configuration. For more information, see the *publish.replication.data.on.startup* and *require.replication.data.on.startup* property descriptions in Cluster management.

Resolved issues

Jetty library upgrade

Fixed PF-31865

We upgraded the Jetty library, resolving CVE-2022-2047 and CVE-2022-2048.

OAuth scope names

Fixed PF-33056

Using `submit` and `onSubmit` as OAuth scope names in the administrative UI drop-down no longer causes front-end JavaScript errors.

Policy fragment validation error

Fixed PF-33156

Policy fragments with valid authentication sources no longer fail with an Invalid Configuration error during runtime.

Eliminating redundant group updates

Fixed PF-33441

PingFederate, when configured with PingDirectory as an outbound provisioning data source, no longer sends redundant group updates in each provisioning cycle when the entry remains unchanged.

Potential security vulnerability

Fixed PF-33449

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

PingFederate as a Windows service

Fixed PF-33450

We fixed an issue so that PingFederate as a Windows service now runs on Java 17. When updating to the latest maintenance release using an in-place update method (for example, from 11.3.0 to 11.3.x), in addition to the steps in Updating to the latest maintenance release, you must remove the existing PingFederate Windows service. After removal, re-install the PingFederate Windows Service to apply this fix.

Authentication policy fail path

Fixed PF-33519

When an OIDC identity provider (IdP) connection fails in an authentication policy, PingFederate now continues on to the fail path of the authentication policy.

Fragment mapping validation error

Fixed PF-33722

We resolved an issue that incorrectly produced an administrative API validation error when the fragment mapping references `context.RequestedUser` as the mapping source.

Authorization details within a RAR

Fixed PF-33863

PingFederate now processes authorization details within a rich authorization request (RAR) as a JSON Array in a JWT request. Additionally, PingFederate no longer supports authorization details sent as stringified JSON arrays.

Cluster engine nodes starting without replication data

Fixed PF-33881

Resolved a replication issue that, in rare cases, caused an engine node in a cluster to start without replication data from other nodes.

Server error when revoking user sessions

Fixed PF-33920

Resolved an issue that prevented user sessions from being revoked through the session management API when using persistent sessions.

Potential security vulnerability

Fixed PF-33935

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Fragment mapping validation errors

Fixed PF-33957

When utilizing the PingFederate administrative API to create or update a fragment that includes another fragment, the API will no longer produce a validation error when fragment mapping involves an input source type.

Updated template variable

Fixed PF-34016

The `message-template-end-user-password-change.html` template now contains the *USERNAME* variable.

Potential security vulnerability

Fixed PF-34017

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Policy evaluation issue

Fixed PF-34051

We fixed a policy evaluation issue that occurred when `ui_locales` was present in an authentication request.

Certificate import improvements

Fixed PF-34074

We updated the administrative UI to include certification serial number in the drop-down, thus preventing import errors for certifications sharing the same Subject DN and expiration date combination.

DynamoDB attribute lookup error

Fixed PF-34099

We fixed an attribute lookup error that occurred when different DynamoDB attributes shared an overlapping path.

PingFederate 11.3 (June 2023)

New features and improvements in PingFederate 11.3.

New features and enhancements

Support for `nbf` and `iat` claims in JWT access token managers

New

Now you can configure access token managers to include the JSON web token (JWT) `access_token` claims `nbf` (not before) and `iat` (issued at). This enables stronger validations by receiving clients or protected resources that process that access_token. For more information, go to Configuring an access token management instance, and in the *JSON web token data model* section click the *JSON token management* tab.

Retries for client-side LDAP errors

New

To further improve reliability and robustness, now PingFederate executes retries rather than failover only. PingFederate initiates a single retry if a request fails and it appears the connection has become invalid. For more information, see the **Retry Failed Operations** field in Setting advanced LDAP options.

Referencing incoming PAR parameters in authentication policies

New

For authorization requests, parameters can now be referenced for incoming PAR requests (pushed authorization requests) inside authentication policies. This lets PingFederate process incoming requests independently of how it received them. For more information, see Pushed authorization requests endpoint.

Unique identifiers for PingFederate transactions

New

To improve logging, PingFederate now uses a `transactionId`. For each transaction, this value won't change between the initial request and the final response. This is especially useful for troubleshooting. For more information, see the `transactionid` field in Security audit logging.

All user attributes available to HTML and mail templates

New

Now you can configure HTML and mail templates with user details. With these details, you can personalize user facing pages and include messages, such as greetings by name, or email addresses that were used for a password recovery flow. The attributes are documented in the templates.

Logging certificate expiration advance warnings

New

Previously, PingFederate produced notifications to inform administrators about expiring certificates. Now you can configure PingFederate to log upcoming expirations without producing notifications. For more information, see Configuring runtime notifications.

Improved European Union compliance with SAML 2.0

New

Two major SAML 2.0 messaging improvements align PingFederate closer to EU regulations:

- Now PingFederate can decrypt `EncryptedID` elements included as SAML attributes. They no longer must be enclosed as an `EncryptedAttribute`. For more information, see Specifying XML encryption policy (for SAML 2.0).
- To enhance signing capabilities, PingFederate now also supports some of the RSASSA-PSS algorithms. For more information, see Signing algorithms.

Support for credential-protected forward proxy servers

New

Because proxy servers can require credentials for authentication purposes, now you can configure PingFederate with proxy server credentials so that connections can be easily established and secured. For more information, see Configuring forward proxy server settings.

Amazon DynamoDB for attribute source lookups

New

Our continued effort to support Amazon DynamoDB (NoSQL) now lets you use DynamoDB as a source for attribute lookups. The connector supports the DynamoDB query language so you can easily configure it. For more information, see Configuring an AWS DynamoDB datastore.

OAuth 2.0 DPoP

New

As regulations for APIs in the context of financial services tighten, it's important to support highly secure API authentication and authorization methods. OAuth DPoP (Demonstrating Proof-of-Possession) is an extension to the OAuth framework and specifies how OAuth tokens are bound to clients. Clients must digitally prove the ownership of these tokens at runtime, which prevents unauthorized clients from misusing them. This extension is useful for any OAuth scenario, not only in financial environments. For more information, see Configuring authorization server settings.

Logging the TLS version that clients use

New

For TLS connections, PingFederate can now log the TLS version that clients use. This gives you an easy way to identify clients that might need updates to use newer versions. For more information, see the tlsversion field in Security audit logging.

Certificate expiration dates added to certificate menus

New

In the administrative console, now certificate selection menus show the distinguished name (DN) and expiration date for each certificate, rather than a serial number. This gives you easy access to relevant information.

New JWT Token Processor

New

A new JWT token processor enhances the token exchange capabilities so that you can leverage any configured issuer. Now PingFederate can validate and accept incoming tokens that were created by pre-configured issuers. For more information, see Configuring a JWT Token Processor 2.0 instance.

Enhanced authentication policies

New

Complex authentication policies are sometimes challenging to manage. To simplify your work and add flexibility to policies, PingFederate provides several policy enhancements:

- Now the Requested AuthN Context Authentication Selector can determine the authentication context for flows. For more information, see Configuring the Requested AuthN Context Authentication Selector.
- Now you can use Context and Extended Properties for attribute sources when mapping authentication policy contracts and local identity profiles. For more information, see Configuring contract mapping, Configuring local identity mapping, and Defining issuance criteria for contract or local identity mapping.
- Now you can use the Scope and Virtual Server ID attributes for authentication sources in policy rules. For more information, see Scope and Virtual Server ID in Configuring rules in authentication policies.
- Now you can use OGNL expressions to configure more complex policy rules. For more information, see Expression in Configuring rules in authentication policies.

PAR support for OIDC IdP connections and OIDC admin authentication

New

PingFederate now initiates outbound authorization requests using the PAR endpoint of the target authorization server if you expose it. This enhancement lets PingFederate use PAR inbound and outbound, which improves OAuth flow security. For more information, see the **Pushed Authorization Request Endpoint** field in Configuring OpenID Provider information.

Ability to include x5t and typ in ID token headers

New

Now PingFederate can include JWT header values `x5t` and `typ` in the ID tokens it issues. You can include the `x5t` header with static keys enabled, whereas you can configure the `typ` header to an appropriate value without a dependency on the types of keys. The `x5t` header adds another mechanism for verifying the validity of a received JWT. For information about the `x5t` and `typ` parameters, see the JSON web key (JWK) and JWT specifications, respectively, and steps 9 and 10 in Configuring policy and ID token settings.

Support for the alg parameter response for JWKS keys

New

The `alg` header is now supported in PingFederate's JWKS endpoint. Any elliptic curve keys and all RSA-256 based keys expose this header. This feature lets clients verify that a received JWT has been signed by the advertised algorithm. For information about the `alg` parameter, see the JWK specification and JSON Web Keys endpoint.

Support for client_secret_jwt as client authentication

New

With the client_secret_jwt authentication method, a client can choose to create a signed JWT when authenticating against PingFederate's token endpoint, introspection endpoint, PAR endpoint, or CIBA endpoint instead of providing the client secret. This feature prevents potential client secret leakage because it's not actively exchanged with any party. PingFederate also supports this feature when it acts as an RP. For more information, see client_secret_jwt in the Open ID Connect specification and Client authentication schemes.

Refresh token reuse and revocation best practice

New

PingFederate now revokes a chain of tokens if a refresh token is revoked or if a refresh token is reused. This includes derived authorization codes and access tokens. For more information, see the Refresh Token settings section of Configuring authorization server settings.

Overriding configuration settings using environment variables

New

Now you can configure many properties as environment variables instead of setting them in properties files. This is especially important for container environments, which is common practice.

Auditing enhancements

New

Several enhancements provide more details in PingFederate generated logs. These include the logging of JWT IDs (jti), hashed values of authorization codes, access tokens, and refresh tokens. Also, PingFederate now logs which system has locked out users after multiple, unsuccessful login attempts, so you'll know if it was PingFederate or an LDAP server. PingFederate also adds more details to the administrative API logs, so now there are almost no differences between logs generated when using the administrative console or administrative API. For more information, see Administrator audit logging, Administrative API audit log, and Security audit logging.

Amazon DynamoDB and OAuth client records

New

Now you can manage OAuth clients in Amazon DynamoDB. With this update, you can use DynamoDB to manage OAuth clients, persistent grants, and persistent authorization sessions. Learn more in Configuring an Amazon DynamoDB for client storage.

Upgraded Velocity Engine 2.3

New

PingFederate now supports Apache Velocity Engine 2.3. For more information, see Upgrading in the Apache Velocity Engine documentation.

Support for strict content security policy (CSP) for HTML templates

New

Now you can include CSP policies for HTML templates without having to implement workarounds. For more information, see Customizable user-facing pages.

Ability to use additional Velocity tools

New

Now you can use Velocity templates with more tools, such as cookieTool.

Support for Microsoft Azure SQL Managed Instance

New

PingFederate now supports Microsoft Azure SQL Managed Instance. For more information, see the Datastore integration table in System requirements, and for more information on how to configure a connection to Microsoft Azure SQL Managed Instance, see Configuring a JDBC connection.

mTLS authentication for REST API datastores

New

PingFederate now supports mutual TLS (mTLS) client authentication for REST API datastores.

mTLS authentication for LDAP datastores

New

PingFederate now supports mTLS client authentication for LDAP datastores.

Entrust nShield Connect HSM and Java 11

New

Now when you integrate an Entrust nShield hardware security module (HSM) with PingFederate, you can use Java 11.

Bundled User Count Utility

New

We added the User Count Utility (UCU) as a bundled component. You can use the UCU to produce unique and active user counts in a PingFederate environment.

Upgraded third-party components

New

We upgraded the following third-party components:

- Upgraded Spring Framework to 5.3.27
- Upgraded jose4j to 0.9.3

Resolved issues

SAML login session tracking

Fixed PF-33168

We improved SP-Initiated SAML login session tracking. This security improvement can affect existing SAML SP connections that rely on multiple session states in a single transaction.

For more information about how your configuration can be affected, and the steps to resolve issues, see Solicited SAML Response Validation in the Ping Identity Support Portal.

Log message when multiple entries match the LDAP PCV search filter

Fixed PF-32427

Now when multiple entries match the LDAP PCV search filter, the following message appears in the log at DEBUG level: `error code 4 - This search operation has sent the maximum of 1 entries to the client`

Multivalued authorization request parameters

Fixed PF-32783

Now multivalued request parameters work as expected in authorization requests for OIDC administrative console authentication.

Tracked parameters in the LDAP search filter when using the administrative API

Fixed PF-32914

Now you can use tracked parameters in the Attribute Sources and User Lookup LDAP search filter when using the administrative API.

Showing and hiding passwords being entered

Fixed PF-33059

Now all password entry fields in PingFederate templates have icons that let users show and hide the password they're entering.

Connections and OAuth clients referencing deleted extended properties

Fixed PF-33311

When a connection or OAuth client references a deleted extended property, PingFederate no longer throws a null pointer exception. Instead it ignores the extended property and logs an error.

Custom error messages from external consent adapters

Fixed PF-33151

Now PingFederate can use customized messages from external consent adapters in error responses.

Restricting password credential validators

Fixed PF-33487

When `restrictToDefaultAccessTokenManager` is enabled on an OAuth client, the client can only get access tokens when being validated by password credential validators that are mapped to the restricted access token manager.

Bypass Authorization Approval and prompt parameters

Fixed PF-33598

When an OAuth client has Bypass Authorization Approval enabled, now that setting takes precedence over the `prompt` parameter in requests.

Document file permissions

Fixed PF-33605

Updated the file permissions of legal documents.

The `memoryoptions` script allocates excessive JVM heap

Fixed PF-33610

The `memoryoptions` script no longer allocates excessive JVM heap on Windows systems.

Authorization Code and Device Authorization grant handling

Fixed PF-33622

For the Device Authorization grant type, if **Check Activation Code** is set to **Before Authentication**, then authorization detail is set in the input parameters map when `IdpAuthenticationAdapterV2` in the SDK is invoked.

Converting the values of binary attributes from PingOne LDAP gateway datastores

Fixed PF-33637

Now when PingFederate retrieves a binary attribute from a PingOne LDAP gateway datastore, it correctly converts the attribute value to the specified format (base64, SID, hex).

Unexpected certificate usage

Fixed PF-33709

When more than one trusted CA matches the issuer DN of an OAuth client, now PingFederate only flags the trusted CA as in use if its certificate hasn't expired and its subject DN matches the client's configured issuer DN.

Potential information disclosure vulnerability

Fixed PF-33867

Removed a potential information disclosure vulnerability.

Jetty unable to serve gzip precompressed resources

Fixed PF-33869

Now PingFederate allows Jetty to precompress resources such as images and CSS.

Returning `400` error instead of a `500` error

Fixed PF-30236

When a system-level issue causes a data source attribute lookup to fail during OAuth flows, if the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.domain.AttributeMapping.xml` file's `AbortOnAttrLookupFailure` attribute is set to `true`, now PingFederate returns a `500` error instead of a `400` error.

Usercount Utility's aggregate command

Fixed PF-32757

When you run the Usercount Utility's aggregate command:

- If all `.ucu` files contain tracking IDs, the utility generates a user count for each event, like before.
- If no `.ucu` files contain tracking IDs, now the utility generates a user count for each application.
- If some `.ucu` files contain tracking ids but others don't:
  - for the files without tracking IDs, now the utility generates a user count for each application.
  - for the files with tracking IDs, now the utility generates a user count for each event.

CPU load displayed as N/A

Fixed PF-32837

Now when the CPU load is 0, heartbeat pages display the value with digits instead of as "N/A".

Unexpected carriage return in audit logs

Fixed PF-32989

We resolved an issue that caused an unexpected carriage return in audit logs during SP-initiated single sign-on (SSO) if an identity provider responded with a non-success status.

Known issues and limitations

PingID password credential validator with integrated RADIUS server

Issue

PingFederate versions 11.1.4, 11.1.5, 11.2.1, and 11.2.2 contain version 3.0.2 of the PingID password credential validator (PCV). That version of the PCV has known issues that you should review before upgrading. For more information, see Known issues in PingID RADIUS PCV 3.0.2.

Administrative console and administrative API

Issue

- Although PingFederate 11.3 and later support DPoP, a known limitation is that the following features don't support DPoP when PingFederate is the RP:
  - The administrative console authentication scheme using OIDC
  - The administrative API authentication scheme using OAuth 2.0
- /bulk: Only resource types currently supported by the administrative API are included in the exported data. We don't intend to introduce administrative API support to the following areas:
  - SAML 2.0 IdP Discovery
  - SAML 2.0 SP Affiliation
  - SMS Provider
- Previously, the administrative API did not accurately reflect a **Persistent Grant Max Lifetime** setting of 29 days (or shorter) with the selection of the **Grants Do Not Timeout Due To Inactivity** option. As a result, if you have configured such OAuth authorization server settings and have generated a bulk export in version 10.0 through 10.0.2, we recommend that you re-generate a new bulk export after upgrading to version 10.0.3 (or a more recent version). The newly exported data does not contain the aforementioned flaw, and you can safely import it to version 10.0.3 (or a more recent version).
- When enabling mTLS certificate-based authentication, administrators often configure a list of acceptable client certificate issuers. When you use a browser to access the console or the administrative API documentation, PingFederate returns to the browser the list of acceptable issuers as part of the TLS handshake. If the browser's client certificate store contains multiple client certificates, the browser often presents you only the certificates whose issuer matches one of the acceptable issuers. However, when PingFederate runs in a Java 11 environment, Chrome presents you all its configured client certificates, regardless of whether the issuer matches one of the acceptable issuers or not.
- When using mTLS authentication to authenticate to an LDAP server for administrative console or administrative API access, PingFederate doesn't support using a Microsoft Active Directory server.
- Prior to toggling the status of a connection with the administrative API, you must ensure that any expired certificates or no longer available attributes are replaced with valid certificates or attributes; otherwise, the update request fails.
- When creating or updating a child instance of a hierarchical plugin, the administrative API retains objects with an `"inherited": false` name/value pair (or without such name/value pair altogether), ignores those with a value of `true`, and returns a 200 HTTP status code. No error messages are returned for the ignored objects.

- Using the browser's navigation mechanisms (for example, the **Back** button) causes inconsistent behavior in the administrative console. Use the navigation buttons provided at the bottom of windows in the PingFederate console.
- Using the PingFederate console in multiple tabs on one browser might cause inconsistent behavior which could corrupt its configuration.
- If authenticated to the PingFederate administrative console using certificate authentication, a session that has timed out might not appear to behave as expected. Normally (when using password authentication), when a session has timed out and a user attempts some action in the console, the browser is redirected to the sign-on page, and then back to the administrative console after authentication is complete. Similar behavior applies for certificate authentication, in principle. However, because the browser might automatically resubmit the certificate for authentication, the browser might redirect to the administrative console and not the sign on page.

PingOne MFA CIBA Authenticator

Issue PingOne MFA

PingFederate 11.3 is not compatible with the PingOne MFA CIBA Authenticator bundled in PingOne MFA Integration Kit version 2.1 and earlier. This issue was resolved in version 2.2 of that integration kit.

TLSv1.3

Issue

For Java versions that don't support TLSv1.3 (meaning versions earlier than 8u261), PingFederate fails on start up with a `NoSuchAlgorithmException` exception. To resolve this error, remove `TLSv1.3` from the following settings in the `run.properties` file:

- `pf.tls.client.protocols`
- `pf.tls.runtime.server.protocols`
- `pf.tls.admin.server.protocols`

TLS cipher suite customization

Issue

PingFederate's TLS cipher suites can be customized by modifying `com.pingidentity.crypto.SunJCEManager.xml` (or a similarly-named file if BCFIPS or an HSM is configured). After updating the file and replicating, all cluster nodes must be restarted for the change to take effect.

Java

Issue

- As of PingFederate 11.1, BC-FIPS and HSMs are not supported when using Java 17.
- Updating Java version 8 to version 11 results in an error when PingFederate is already installed and running on Windows. To work around this issue, uninstall and reinstall the PingFederate Windows service by running the `UninstallPingFederateService.bat` and `InstallPingFederateService.bat` files located in `<pf_install>/pingfederate/sbin/wrapper`.

HSMs

Issue

AWS CloudHSM

- It is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- TLS 1.3 is not currently supported.

Thales HSMs

- JWT token decryption using ECDH-ES may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored on the HSM, and PingFederate is consuming a token encrypted with this key.
- It is not possible to use an EC certificate as an SSL server certificate.
- TLS 1.3 is not currently supported.

Entrust HSMs

- JWT token decryption using ECDH-ES may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored on the HSM, and PingFederate is consuming a token encrypted with this key.
- It is not possible to import a PKCS12- or PEM-formatted EC certificate.
- It is not possible to use an EC certificate as an SSL server certificate.
- TLS 1.3 is not currently supported.

SSO and SLO

Issue

- When consuming SAML metadata, PingFederate does not report an error when neither the `validUntil` nor the `cacheDuration` attribute is included in the metadata. Note that PingFederate does reject expired SAML metadata as indicated by the `validUntil` attribute value, if it is provided.
- The anchored-certificate trust model cannot be used with the Single log off (SLO) redirect binding because the certificate cannot be included with the logout request.
- If an IdP connection is configured for multiple virtual server IDs, PingFederate will always use the default virtual server ID for IdP Discovery during an SP-initiated SSO event.

Composite Adapter configuration

Issue

SLO is not supported when users are authenticated through a Composite Adapter instance that contains another instance of the Composite Adapter.

Self-service password reset

Issue

Passwords can be reset for Microsoft Active Directory user accounts without the permission to change password.

Session revocation API

Issue PPQ-33519

POST requests to the Session Revocation API do not support the Private Key JWT authentication type.

OAuth

Issue

PingFederate does not support a case-sensitive naming convention for OAuth client ID values when client records are stored in a directory server. For example, after creating a client with an ID value of `sampleClient`, PingFederate does not allow the creation of another client with an ID value of `SampleClient`.

Although it's possible to create clients using the same ID values with different casings when client records are stored in XML files, a database server, or custom storage, we recommend not doing so to avoid potential record migration issues.

Customer identity and access management

Issue

Some browsers display a date-picker user interface for fields that have been designed for date-specific inputs. Some browsers do not. If one or more date-specific fields are defined on the registration page or the profile management page (or both), end users must enter the dates manually if their browsers do not display a date-picker user interface for those fields.

Provisioning

Issue

- LDAP referrals return an error and cause provisioning to fail if the `user` or `group` objects are defined at the DC level, and not within an OU or within the Users CN.
- The `totalResults` value in SCIM responses indicates the number of results returned in the current response, not the total number of estimated results on the LDAP server.

Logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

Database logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

RADIUS NAS-IP-Address

Issue

The RADIUS NAS-IP-Address is only included in Access-Request packets when the `pf.bind.engine.address` is set with an IPv4 address. IPv6 is not supported.

Amazon SNS Notification Publisher

Issue

When deploying PingFederate with a forward proxy, plugins based on the AWS SDK, such as the Amazon SNS Notification Publisher, will only honor the `http.proxyHost`, `http.proxyPort`, `http.proxyUser`, and `http.proxyPassword` properties in `run.properties`. The plugin will rely on these properties even if the service URL is `https`.

PingOne Fraud integration kit

Info PingOne Fraud

The PingOne Fraud integration kit is no longer bundled with PingFederate.

Deprecated features

Microsoft Internet Explorer 11

Info

Ping Identity commits to deliver the best experience for administrators and users. As we continue to improve our products, we encourage you to migrate off of Microsoft Internet Explorer 11. Starting with PingFederate 11.0, Internet Explorer 11 is no longer included in the PingFederate qualification process for administrators or users. For a list of supported browsers, see System requirements.

Configcopy tool, Connection Management Service, SSO Directory Service

Info

As of PingFederate 10.2, these features have been deprecated and will be removed in a future release.

Oracle Directory Server Enterprise Edition

Info

As Oracle ended its Premier Support for Oracle Directory Server Enterprise Edition (ODSEE 11g) in December 2019, we no longer include ODSEE as part of the PingFederate qualification process (starting with PingFederate 10.2). We continue to qualify against Oracle Unified Directory (www.oracle.com/middleware/technologies/unified-directory.html) and other supported directory servers. For a full list, see System requirements.

SNMP

Info

Starting with PingFederate 10.2, monitoring and reporting through the SNMP has been removed.

Roles and protocols

Info

Starting with PingFederate 10.1, roles and protocols are always enabled and no longer configurable through the administrative console and API.

S3_PING discovery protocol

Info

Starting with PingFederate 10.1, the S3_PING discovery protocol has been deprecated. Customers running on AWS infrastructure should instead use NATIVE_S3_PING.

Red Hat Enterprise Linux install script

Info

Starting with PingFederate 10.0, the Red Hat Enterprise Linux install script is no longer available. To install PingFederate 10.0 for Linux, you must download and extract the product distribution `.zip` file.

PingFederate 11.2.11 (December 2024)

Resolved issues

Cross-site scripting

Security PF-36304 PF-36311 PF-36313

We've fixed a security vulnerability where PingFederate accepted cross-site scripting inputs.

Email verification failure after registration workflow

Fixed PF-36574

We've fixed a defect that caused the email verification screen to fail to appear when a user registered through an authentication source.

PingFederate 11.2.10 (July 2024)

Resolved issues

OAuth client only validates one access token manager when `aud` parameter included

Fixed PF-35737

Fixed a defect that caused PingFederate to validate only the first OAuth client access token manager it found when **Validate Against All Eligible Access Token Managers** was checked, and the `aud` parameter was included in the request.

Wrong content-type for autopost template form

Fixed PF-35784

Fixed a defect that caused PingFederate to send the wrong content-type value for `form.autopost.template.html` requests, which caused the page to render as JSON rather than the formatted form.

PingFederate 11.2.9 (April 2024)

Resolved issues

Rest datastore security vulnerability

Security PF-34720

Fixed a JSON injection vulnerability in REST datastores described in security advisory SECADV044.

Runtime nodes security vulnerability

Security PF-34896

Fixed a path traversal vulnerability in Runtime nodes described in security advisory SECADV044.

OpenID Connect policy management editor security vulnerability

Security PF-35081

Fixed a Cross-Site Scripting vulnerability in the OpenID Connect Policy Management Editor described in security advisory SECADV044.

Slow log consumption affects performance

Fixed PF-33368

Fixed a defect that caused performance issues for PingFederate when third-party logging services were slow to consume logging events.

RHEL 8 using OS-level FIPS causes PingFederate failure

Fixed PF-34879

Fixed a defect that caused PingFederate to fail on startup when installed on a Red Hat Enterprise Linux (RHEL) server with OS-levels FIPS enabled.

PingFederate 11.2.8 (December 2023)

New features and enhancements

Configurable option to turn on/off plugin creation and initialization during PingFederate startup.

New PF-34640

Added the *ConfigurePluginsOnStartup* variable to the `config-store` file.

Default value of `true` creates and initializes plugins during startup. `false` prevents creation and initialization of plugins, which can reduce startup time.

Improved OGNL expression logging

Improved PF-34050

The administrator audit log file (`admin.log`) now logs any OGNL expression tests performed and the expression variables used with an event type of `TEST_EXPRESSION` .

For more information, see Administrator audit logging

Resolved issues

Resolved a vulnerability in the Initial Setup Wizard

Security PF-34646

Fixed a Server-Side Request Forgery vulnerability in the Initial Setup Wizard described in security advisory SECADV041.

PingFederate systematically adds server-side sort control

Fixed PF-33466

You can now turn off server-side sorting using a configuration option.

Updating OAuth clients with dynamic client registration

Fixed PF-34146

Fixed a defect where an OAuth client created with dynamic client registration (DCR) couldn't be updated with DCR after it was modified with the administrative console.

Unable to deobfuscate chunked grant value with character length of 682

Fixed PF-34839

Fixed a defect where PingFederate was unable to deobfuscate grant attributes for a small group of users in OAuth flows.

PingFederate 11.2.7 (August 2023)

Resolved issues

Potential security vulnerability

Fixed PF-33449

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Potential security vulnerability

Fixed PF-34017

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Policy evaluation issue

Fixed PF-34051

We fixed a policy evaluation issue that occurred when `ui_locales` was present in an authentication request.

PingFederate 11.2.6 (June 2023)

Resolved issues

Policy fragment validation error

Fixed PF-33156

Policy fragments with valid authentication sources no longer fail with an Invalid Configuration error during runtime.

The `memoryoptions` utility allocates excessive JVM heap

Fixed PF-33610

The `memoryoptions` utility no longer allocates excessive JVM heap on Windows systems.

The device authorization grant type and the Check Activation Code setting

Fixed PF-33622

For the device authorization grant type, if **Check Activation Code** is set to **Before Authentication**, now authorization detail is set in the input parameters map when `IdpAuthenticationAdapterV2` in the SDK is invoked.

Binary attributes handled incorrectly when using PingOne LDAP gateway datastores

Fixed PF-33637

Now when PingFederate retrieves a binary attribute from a PingOne LDAP gateway datastore, it correctly converts the attribute value to the specified format (base64, SID, hex).

Jetty unable to precompress resources

Fixed PF-33869

Now PingFederate allows Jetty to precompress resources such as images and CSS.

Cluster engine nodes starting without replication data

Fixed PF-33881

Resolved a replication issue that, in rare cases, caused an engine node in a cluster to start without replication data from other nodes.

Server error when revoking user sessions

Fixed PF-33920

Resolved an issue that prevented user sessions from being revoked through the session management API when using persistent sessions.

PingFederate 11.2.5 (May 2023)

Resolved issues

Logging validation

Fixed PF-32764

We've improved logging validation.

Multi-value request parameters for OIDC for console login

Fixed PF-32783

We fixed an issue where multi-value request parameters were not working as expected when using OIDC for console login.

Preservation of changes to certain validation rules

Fixed PF-33093

We fixed an issue where PingFederate did not preserve changes to certain validation rules in the `http-request-parameter-validation.xml` file upon upgrade.

SAML login session tracking

Fixed PF-33168

We improved SP-Initiated SAML login session tracking. This security improvement can affect existing SAML SP connections that rely on multiple session states in a single transaction.

For more information about how your configuration can be affected, and the steps to resolve issues, see Solicited SAML Response Validation in the Ping Identity Support Portal.

OTL reset page error messaging

Fixed PF-33307

The one-time link (OTL) reset page now displays an error message when the link is expired.

Access token bug fix

Fixed PF-33342

We resolved an issue where an access token may not include the `pi.sri` claim after refresh. This issue only occurs when reuse of existing access grants is enabled.

Attribute retrieval

Fixed PF-33484

In OAuth and OpenID Connect (OIDC) flows, external consent adapters can now retrieve attributes from the chained attributes map.

LDAP bug fix

Fixed PF-33503

We fixed an LDAP issue where new access grant records were not created with new scopes when **Reuse Existing Persistent Access Grants for Grant Types** was enabled.

ID token ACR claim

Fixed PF-33557

We resolved an issue where an ID token would not include the Authentication Context Class Reference (ACR) claim if an old client secret was used during the retention period.

Redundancies in key algorithm generation

Fixed PF-33607

We fixed an issue that affected cluster replication when PingFederate was deployed with AWS CloudHSM. When replication was initiated, engines generated a number of temporary key pairs, and the increased load on the HSM could trigger SSO errors.

PingFederate 11.2.4 (March 2023)

Resolved issues

Base DN reference attribute

Fixed PF-32971

We've fixed an issue where upstream data stores in the chain could not recognize the reference attribute for base distinguished name (DN) during lookup.

Identity store provisioner validation rules

Fixed PF-33017

We've improved validation rules to ensure that only identity store provisioners that support groups require group attribute contract validation.

DCR with client secret retention

Fixed PF-33035

We've resolved a null pointer exception (NPE) error that occurred when attempting to set up Dynamic Client Registration (DCR) with client secret retention.

Response handling logs

Fixed PF-33131

We've added additional logging to help debug unexpected errors in response handling.

OIDC policies with fragments configured

Fixed PF-33158

We've resolved an issue that occurred when saving OpenID Connect (OIDC) policies with fragments configured.

LDAP filter validation

Fixed PF-33173

We've fixed an issue related to LDAP filter validation that arose when accessing the **IdP Connections** page.

HSM key sessions

Fixed PF-33284

We've improved the process of cluster replication for PingFederate integrated with AWS CloudHSM by purging HSM key pair generation sessions.

PingFederate 11.2.3 (February 2023)

Resolved issues

Potential security vulnerability

Fixed PF-32748

We've resolved a potential security vulnerability that is described in security advisory SECADV035.

Log improvements

Fixed PF-33017

In order to reduce re-encryption and file scanning log verbosity, when a configuration is imported or replicated to a cluster, PingFederate no longer scans files in the `etc` directory.

Other improvements

New

- We also updated the following bundled components and third-party dependencies:
  - PingID Integration Kit 2.24
  - PingID Adapter 2.13.2
  - PingID PCV (with integrated RADIUS server) 3.0.3

PingFederate 11.2.2 (February 2023)

Resolved issues

Server log warnings

Fixed PF-33037

We've added a warning to server logs if the *ds-pwp-state-json* attribute is not present in PingDirectory's LDAP Response. This warning appears in the log every time a user interacts with the profile management page. Please enable this attribute to adhere to PingDirectory's security configuration best practices. PingDirectory version 8.1 and later supports this attribute, and customers running older versions are encouraged to upgrade to a supported version as soon as possible.

PingFederate 11.2.1 (February 2023)

Resolved issues

OAuth client management

Fixed PF-32790

When managing OAuth clients, we've resolved a defect where selecting the **Require JWT Secured Authorization Response Mode** text toggled the incorrect check box.

Potential security vulnerability

Fixed PF-32805

We've resolved a potential security vulnerability that is described in security advisory SECADV033.

Informing adapters of end policy result

Fixed PF-32890

When processing policy fragments, all adapters invoked in the fragment now correctly execute their respective post-processing step (if applicable) to inform the adapter of the end policy result.

Managing certificates within Metadata Export

Fixed PF-32965

Managing certificates within the **Metadata Export** flow no longer displays or saves an empty list of certificates, clearing out existing ones in the process. For more information, see Metadata export.

Cluster data replication

Fixed PF-32983

We've resolved a defect where cluster data replication could remove keys from engine node's `pf.jwk` file instead of merging and retaining the keys.

Other improvements

New

- We also updated the following bundled components and third-party dependencies:
    - PingID Integration Kit 2.23
    - PingID Adapter 2.13.2
    - PingID PCV (with integrated RADIUS server) 3.0.2

> **Note**
>
> This version of the PingID PCV has known issues that you should review before upgrading. For more information, see Known issues in PingID RADIUS PCV 3.0.2.

PingFederate 11.2 (December 2022)

New features and improvements in PingFederate 11.2.

New features and enhancements

Support for OAuth 2.0 authorization server metadata

New

PingFederate now supports OAuth 2.0 authorization server metadata. This allows OAuth clients to retrieve relevant endpoints and other details about features that PingFederate supports. The API response is like the OpenID Connect Discovery endpoints response but doesn't include OpenID Connect relevant details. This lets you configure endpoints for your particular use case. See OAuth authorization server metadata endpoint.

Support for nested groups and nested search for PingDirectory

New

For outbound provisioning, PingFederate now supports nested groups and nested search for PingDirectory. This lets you freely choose your favorite directory without needing to choose based on the support for nested groups. See nested group and nested search in Specifying a source location.

Exposed `AccessGrantManagerAccessor` as part of the SDK

New

The `AccessGrantManagerAccessor` is now accessible in the PingFederate SDK. This lets developers query existing persistent grants at run time. See `<pf_install>/pingfederate/sdk/doc/com/pingidentity/access/AccessGrantManagerAccessor.html` in the SDK documentation.

Improved the sign-on experience after users change their password

New

Now you can configure PingFederate to keep users signed in after they change their password. This prevents users from having to sign on again, right after updating their password, improving the user experience. See the **Require Re-authentication** settings HTML Form Adapter advanced fields.

Administrative API supports multiple authentication and authorization schemes

New

Now you can configure the PingFederate administrative API to accept either OAuth `access_token` or basic authentication. This is especially useful in cases where applications shouldn't include administrator's credentials in API requests. See `pf.admin.api.authentication` in Configuring PingFederate properties.

Support for Google reCAPTCHA v3 and integration with multiple CAPTCHA providers

New

PingFederate now supports Google reCAPTCHA v3. reCAPTCHA v3 produces a score between 0.0 - 1.0 (risky to safe) that you can use in policies to require step-up authentication or other actions. By default, reCAPTCHA v3 doesn't interrupt user journeys, which are in the control of application developers. Learn more in Managing CAPTCHA and risk providers.

PingFederate also now provides an SDK that allows for integrations with custom CAPTCHA providers, which adds great flexibility to the CAPTCHA feature.

Improved cluster replication notification

New

Instead of showing an active bell icon, the administrative console now displays a banner when cluster replication is required. The banner includes a link to the **Cluster Management** window for easy access. See Cluster management.

The administrative console supports OIDC claims parameter

New

You can configure PingFederate to function as an OpenID Connect client and let administrators sign on to the administrative console using their PingOne credentials. PingFederate initiates an OpenID Connect flow that includes the claims parameter. You can also use this feature outside the PingOne environment, leveraging any authorization server that supports the claims parameter. This allows for a simpler, seamless login flow. See Request Parameters in Enabling OIDC-based authentication.

The administrative console supports third party-initiated login

New

You can configure PingFederate to accept incoming parameters, such as `iss`, that are processed and included in an outgoing authorization request if configured to do so. This feature lets administrators sign on to PingFederate from PingOne. This feature also supports other OpenID Connect authorization servers that support incoming parameters. See Request Parameters in Enabling OIDC-based authentication.

PingOne DaVinci integration kit

New

The PingFederate distribution now includes the PingOne DaVinci integration kit. See PingOne DaVinci Adapter in Bundled adapters and authenticators.

Amazon DynamoDB and persistent authentication sessions

New

PingFederate can now manage persistent user sessions in AWS DynamoDB. Persistent user sessions keep sessions active even after a restart of PingFederate. This feature reduces the interruption of user journeys. See *Configuring an Amazon DynamoDB for persistent authentication sessions* in Defining a datastore for persistent authentication sessions.

Enhanced policy rules

New

When defining policy rules, now attributes that were processed in an earlier step can be accessed further down in the policy tree. This feature enhances the management and usability of policies. See Configuring rules in authentication policies.

The heartbeat endpoint and JMX expose more information

New

The data exposed by the heartbeat endpoint and JMX interface now include more details, such as the number of errors per data store. See Liveliness and responsiveness.

Updated the bundled PingOne MFA Adapter

Improved

Updated the bundled PingOne MFA Adapter to the newest version, 2.0. See PingOne MFA Adapter in Bundled adapters and authenticators.

Toggle log verbosity with ease

New

Gone are the days you had to edit the `log4j2.xml` file on multiple servers to enable or disable DEBUG messages in their server logs. Now you can toggle log settings in the administrative console or with the administrative API.

PingFederate provides a set of message categories, each targeting a specific scenario. For example, the **XML Signatures** category helps you troubleshoot XML signature issues. You can also add your own categories to suit your unique requirements.

Timestamps for clients and connections

New

When viewing lists of OAuth clients and Browser single sign-on (SSO)/security token service (STS) connections, you can now sort them by modification or creation time. The timestamps can also help you understand the history and the relationship between clients and connections.

AWS CloudHSM and Java 11

New

If you integrate with Amazon Web Services (AWS) CloudHSM, now you can choose between Java 8 and Java 11.

OAuth Rich Authorization Requests

New

OAuth rich authorization requests (RAR) provide a standard way for OAuth client applications to specify fine-grained authorization requirements in their requests. For example, when initiating a money transfer, a personal banking application can pass all relevant information to the authorization server via the new parameter `authorization_details`. The authorization server supporting RAR processes the `authorization_details` parameter value accordingly and ultimately returns tokens to the application if the process completes successfully.

RAR is on track to become a requirement in Financial-grade API (FAPI) 2.0. With this new capability, you can confidently build your open banking solutions with PingFederate.

Other enhancements

New

Now you can optionally define a sender name for each SMTP notification publisher instance.

PingFederate now supports XML Encryption 1.1.

Resolved issues

Sorting LDAP and database-related fields

Fixed PF-29355

For LDAP and database-related fields, PingFederate now sorts values alphabetically and in case-insensitive order.

Detailed comments added to log4j2.xml file

Fixed PF-30514

We've added detailed comments to the `log4j2.xml` file to prevent misconfigurations that could lead to service hangs and production outages. For more information on logging, see Log4j 2 logging service and configuration.

Configuration options added to control SAML error responses

Fixed PF-30514

We've added a configuration option to control whether SAML error responses include `Cause`. The new setting is `IncludeErrorCauseInSamlResponse` in `config-store/org.sourceid.saml20.protocol.StatusResponseTypeUtil.xml`. The default value is *true*.

Improved SP STS message customization

Fixed PF-31149

The *#HttpServletRequest* and *#HttpServletResponse* variables are now available in SP STS message customization. For more information, see Message types and available variables.

Connections with multiple protocol types

Fixed PF-31531

We've resolved an issue where connections with multiple protocol types would only filter on a single protocol type.

OpenID Connect (OIDC) for administrative console authentication

Fixed PF-31717

When using OIDC for administrative console authentication, PingFederate no longer throws an NPE if `private_key_jwt` is used for client authentication method and the `client.secret` property is not set.

Improvements to refresh token rolling criteria

Fixed PF-31761

We've introduced a new separate stored value to track when refresh tokens should be reissued to OAuth clients, resolving a defect where rolling refresh tokens read the incorrect update timestamp to determine refresh token rolling criteria. For more information, see Configuring authorization server settings.

Store clients with special characters

Fixed PF-31786

When adding clients to Active Directory (AD) or other LDAP stores, PingFederate now automatically escapes reserved characters from clientIDs.

Improved detection around invalid Group DN

Fixed PF-31791

We've improved detection around invalid Group distinguished names (DN) and added exceptions in the provisioner log. For more information on Group DN, see Specifying a source location.

Updates to the SameSite=None header attribute supported browsers list

Fixed PF-31806

We've updated the supported browsers list for the `SameSite=None` header attribute to filter out problematic clients with the `SameSite` cookie attribute bug: Safari version 12 and Embedded Apple Webkit Browser Safari 12 on macOS.

Expired user sessions and session log out

Fixed PF-31807

PingFederate's administrative console now identifies expired user sessions on timeout and properly removes the session regardless of user interaction.

Policy and fragment logging

Fixed PF-31862

PingFederate now logs the policy and fragment name before fragment processing.

Bulk import for IdP connections

Fixed PF-31870

Resolved an issue where bulk import fails for identity provider (IdP) connections that fulfill Persistent Grant Extended Attributes.

Template double-submission

Fixed PF-31957

PingFederate templates no longer allow double-submission.

Connection failures on external LDAP authentication login

Fixed PF-32001

PingFederate now recovers from initial connection failure when logging into the administrative console using external LDAP authentication.

Hiding user information from authentication API responses

Fixed PF-32028

You can now configure the `IncludeUserInfoInResponses` setting in the `<install dir>/server/default/data/config-store/org.sourceid.saml20.domain.mgmt.impl.AuthnApiManagerImpl.xml` file to hide user information from authentication API responses.

Errors on policy fragments configured to handle failures locally

Fixed PF-32073

When an error occurs on policies containing fragments and configured to handle failures locally, PingFederate no longer redirects a user to the service provider (SP) error page on SP-initiated SSO.

Password management

Fixed PF-32081

We've resolved an issue around password requirements messaging during password management.

Updated description text on Import Connections page

Fixed PF-32088

We've updated the description text on the import IdP/SP connection page to indicate that PingFederate only performs minimal validation for imported connections. We suggest using the administrative API for connection migration, which performs thorough validation.

OTL for password reset expiry or reuse error reporting

Fixed PF-32090

In the case where a one-time link (OTL) for password reset expires or is reused, PingFederate now responds with the appropriate error message in the authentication API and logs the error response in the `audit.log`. For more information on OTL for password reset, see Configuring self-service account recovery.

Duplicate scope and scope group name values

Fixed PF-32234

We've resolved a defect that allowed scope and scope group names to be the same when saved through the administrative console. For more information on scopes, see Scopes and scope management.

Warning during SQL provisioning table creation

Fixed PF-32254

We've decreased the maximum key length for `saasGroupName`, resolving a warning that occurred when creating SQL provisioning tables.

'Change Password' link accessibility

Fixed PF-32343

On sign-on pages, we've improved the accessibility of the 'Change Password' link, regardless of browser window size.

Notification publisher accessor added to SDK

Fixed PF-32345

We've added a notification publisher accessor to the SDK, addressing an error where plugins utilizing a notification publisher could not invoke one of the notification publishers configured in PingFederate.

Fragment processing now independent of policy processing

Fixed PF-32461

PingFederate now processes policy fragments independently from policies and other fragments.

LIP registration via a third-party service and the authentication API

Fixed PF-32574

We've resolved a defect where Local Identity Profile (LIP) registration via a third-party service and the authentication API would still require a password, despite previously registering with the third party.

Known issues and limitations

PingID password credential validator with integrated RADIUS server

Issue

PingFederate versions 11.1.4, 11.1.5, 11.2.1, and 11.2.2 contain version 3.0.2 of the PingID password credential validator (PCV). That version of the PCV has known issues that you should review before upgrading. For more information, see Known issues in PingID RADIUS PCV 3.0.2.

Administrative console and administrative API

Issue

- /bulk: Only resource types currently supported by the administrative API are included in the exported data. We don’t intend to introduce administrative API support to the following areas:
  - SAML 2.0 IdP Discovery
  - SAML 2.0 SP Affiliation
  - SMS Provider
- Previously, the administrative API did not accurately reflect a **Persistent Grant Max Lifetime** setting of 29 days (or shorter) with the selection of the **Grants Do Not Timeout Due To Inactivity** option. As a result, if you have configured such OAuth authorization server settings and have generated a bulk export in version 10.0 through 10.0.2, we recommend that you re-generate a new bulk export after upgrading to version 10.0.3 (or a more recent version). The newly exported data does not contain the aforementioned flaw, and you can safely import it to version 10.0.3 (or a more recent version).
- When enabling mutual TLS certificate-based authentication, administrators often configure a list of acceptable client certificate issuers. When you use a browser to access the console or the administrative API documentation, PingFederate returns to the browser the list of acceptable issuers as part of the TLS handshake. If the browser’s client certificate store contains multiple client certificates, the browser often presents you only the certificates whose issuer matches one of the acceptable issuers. However, when PingFederate runs in a Java 11 environment, Chrome presents you all its configured client certificates, regardless of whether the issuer matches one of the acceptable issuers or not.
- Prior to toggling the status of a connection with the administrative API, you must ensure that any expired certificates or no longer available attributes are replaced with valid certificates or attributes; otherwise, the update request fails.
- When creating or updating a child instance of a hierarchical plugin, the administrative API retains objects with an `"inherited": false` name/value pair (or without such name/value pair altogether), ignores those with a value of `true`, and returns a 200 HTTP status code. No error messages are returned for the ignored objects.
- Using the browser’s navigation mechanisms (for example, the **Back** button) causes inconsistent behavior in the administrative console. Use the navigation buttons provided at the bottom of windows in the PingFederate console.
- Using the PingFederate console in multiple tabs on one browser might cause inconsistent behavior which could corrupt its configuration.

- If authenticated to the PingFederate administrative console using certificate authentication, a session that has timed out might not appear to behave as expected. Normally (when using password authentication), when a session has timed out and a user attempts some action in the console, the browser is redirected to the sign-on page, and then back to the administrative console after authentication is complete. Similar behavior applies for certificate authentication, in principle. However, because the browser might automatically resubmit the certificate for authentication, the browser might redirect to the administrative console and not the sign on page.

TLSv1.3

Issue

For Java versions that don't support TLSv1.3 (meaning versions earlier than 8u261), PingFederate fails on start up with a `NoSuchAlgorithmException` exception. To resolve this error, remove `TLSv1.3` from the following settings in the `run.properties` file:

- `pf.tls.client.protocols`
- `pf.tls.runtime.server.protocols`
- `pf.tls.admin.server.protocols`

TLS cipher suite customization

Issue

PingFederate's TLS cipher suites can be customized by modifying `com.pingidentity.crypto.SunJCEManager.xml` (or a similarly-named file if BCFIPS or a hardware security module (HSM) is configured). After updating the file and replicating, all cluster nodes must be restarted for the change to take effect.

Java

Issue

- As of PingFederate 11.1, BC-FIPS and HSMs are not supported when using Java 17.
- Updating Java version 8 to version 11 results in an error when PingFederate is already installed and running. To work around this issue, uninstall and reinstall the PingFederate Windows service by running the `UninstallPingFederateService.bat` and `InstallPingFederateService.bat` files located in `<pf_install>/pingfederate/sbin/wrapper`.

Hardware security modules (HSMs)

Issue

AWS CloudHSM

- It is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- TLS 1.3 is not currently supported.

Thales HSMs

- JWT token decryption using ECDH-ES may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored on the HSM, and PingFederate is consuming a token encrypted with this key.
- It is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.

- TLS 1.3 is not currently supported.

Entrust HSMs

- PingFederate must be deployed with Oracle Server Java Runtime Environment (JRE) 8 or Amazon Corretto 8.
- JWT token decryption using ECDH-ES or RSAES OAEP may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored on the HSM, and PingFederate is consuming a token encrypted with this key.
- SAML assertion decryption using RSA OAEP may fail when the decryption key is stored on the HSM.
- It is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- TLS 1.3 is not currently supported.

SSO and SLO

Issue

- When consuming SAML metadata, PingFederate does not report an error when neither the `validUntil` nor the `cacheDuration` attribute is included in the metadata. Note that PingFederate does reject expired SAML metadata as indicated by the `validUntil` attribute value, if it is provided.
- The anchored-certificate trust model cannot be used with the Single log off (SLO) redirect binding because the certificate cannot be included with the logout request.
- If an IdP connection is configured for multiple virtual server IDs, PingFederate will always use the default virtual server ID for IdP Discovery during an SP-initiated SSO event.

Composite Adapter configuration

Issue

SLO is not supported when users are authenticated through a Composite Adapter instance that contains another instance of the Composite Adapter.

Self-service password reset

Issue

Passwords can be reset for Microsoft Active Directory user accounts without the permission to change password.

OAuth

Issue

PingFederate does not support a case-sensitive naming convention for OAuth client ID values when client records are stored in a directory server. For example, after creating a client with an ID value of `sampleClient`, PingFederate does not allow the creation of another client with an ID value of `SampleClient`.

Although it's possible to create clients using the same ID values with different casings when client records are stored in XML files, a database server, or custom storage, we recommend not doing so to avoid potential record migration issues.

Customer identity and access management

Issue

Some browsers display a date-picker user interface for fields that have been designed for date-specific inputs. Some browsers do not. If one or more date-specific fields are defined on the registration page or the profile management page (or both), end users must enter the dates manually if their browsers do not display a date-picker user interface for those fields.

Provisioning

Issue

- LDAP referrals return an error and cause provisioning to fail if the `user` or `group` objects are defined at the DC level, and not within an OU or within the Users CN.
- The `totalResults` value in SCIM responses indicates the number of results returned in the current response, not the total number of estimated results on the LDAP server.

Logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

Database logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

RADIUS NAS-IP-Address

Issue

The RADIUS NAS-IP-Address is only included in Access-Request packets when the `pf.bind.engine.address` is set with an IPv4 address. IPv6 is not supported.

PingOne Fraud integration kit

Info PingOne Fraud

The PingOne Fraud integration kit is no longer bundled with PingFederate.

Deprecated features

Microsoft Internet Explorer 11

Info

Ping Identity commits to deliver the best experience for administrators and users. As we continue to improve our products, we encourage you to migrate off of Microsoft Internet Explorer 11. Starting with PingFederate 11.0, Internet Explorer 11 is no longer included in the PingFederate qualification process for administrators or users. For a list of supported browsers, see System requirements.

Configcopy tool, Connection Management Service, SSO Directory Service

Info

As of PingFederate 10.2, these features have been deprecated and will be removed in a future release.

Oracle Directory Server Enterprise Edition

Info

As Oracle ended its Premier Support for Oracle Directory Server Enterprise Edition (ODSEE 11g) in December 2019, we no longer include ODSEE as part of the PingFederate qualification process (starting with PingFederate 10.2). We continue to qualify against Oracle Unified Directory (www.oracle.com/middleware/technologies/unified-directory.html) and other supported directory servers. For a full list, see System requirements.

SNMP

Info

Starting with PingFederate 10.2, monitoring and reporting through the Simple Network Management Protocol (SNMP) has been removed.

Roles and protocols

Info

Starting with PingFederate 10.1, roles and protocols are always enabled and no longer configurable through the administrative console and API.

S3_PING discovery protocol

Info

Starting with PingFederate 10.1, the S3_PING discovery protocol has been deprecated. Customers running on AWS infrastructure should instead use NATIVE_S3_PING.

Red Hat Enterprise Linux install script

Info

Starting with PingFederate 10.0, the Red Hat Enterprise Linux install script is no longer available. To install PingFederate 10.0 for Linux, you must download and extract the product distribution `.zip` file.

PingFederate 11.1.11 (January 2025)

Resolved issues

Eliminating redundant group updates

Fixed PF-33441

We've fixed a defect that caused PingFederate, when configured with PingDirectory as an outbound provisioning data source, to send redundant group updates in each provisioning cycle when the entry remains unchanged.

Provisioner uses the wrong time zone when data source and PingFederate are in different time zones

Fixed PF-35286

We've fixed a defect that caused redundant user provisioner updates when the data source and PingFederate were in different time zones.

Group membership loss during provisioning

Fixed PF-36874

We've fixed a defect that caused PingFederate to lose user group membership information when it lost contact with the data store during provisioning operations.

PingFederate 11.1.10 (April 2024)

Resolved issues

Rest datastore security vulnerability

Security PF-34720

Fixed a JSON injection vulnerability in REST datastores described in security advisory SECADV044.

Runtime nodes security vulnerability

Security PF-34896

Fixed a path traversal vulnerability in Runtime nodes described in security advisory SECADV044.

OpenID Connect policy management editor security vulnerability

Security PF-35081

Fixed a Cross-Site Scripting vulnerability in the OpenID Connect Policy Management Editor described in security advisory SECADV044.

Slow log consumption affects performance

Fixed PF-33368

Fixed a defect that caused performance issues for PingFederate when third-party logging services were slow to consume logging events.

PingFederate 11.1.9 (November 30)

Resolved issues

Outbound provisioning performance improvement

Fixed PF-33466

You can now turn off server-side sorting for LDAP requests related to outbound provisioning, which can improve performance in some environments.

Configure this option using the `ProvisionWithServerSort` parameter in the `com.pingidentity.common.util.ldap.LDAPUtil.xml` file.

Updating OAuth clients with dynamic client registration

Fixed PF-34146

Fixed a defect where an OAuth client created with dynamic client registration (DCR) couldn't be updated with DCR after it was modified with the administrative console.

Resolved a vulnerability in the Initial Setup Wizard

Security PF-34646

Fixed a Server-Side Request Forgery vulnerability in the Initial Setup Wizard described in security advisory SECADV041.

PingFederate 11.1.8 (August 2023)

Resolved issues

Potential security vulnerability

Fixed PF-33449

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Binary attributes handled incorrectly when using PingOne LDAP gateway datastores

Fixed PF-33637

Now when PingFederate retrieves a binary attribute from a PingOne LDAP gateway datastore, it correctly converts the attribute value to the specified format (base64, SID, hex).

Potential security vulnerability

Fixed PF-34017

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

PingFederate 11.1.7 (May 2023)

Resolved issues

Logging validation

Fixed PF-32764

We've improved logging validation.

Resource Owner (RO) Password Credentials flow

Fixed PF-33359

We've improved the error messaging around the Resource Owner (RO) Password Credentials flow.

Requested Authentication Context Selector

Fixed PF-33549

The Requested Authentication Context Selector no longer throws a Null Pointer Exception (NPE) during callback.

PingFederate 11.1.6 (February 2023)

Resolved issues

Log improvements

Fixed PF-33017

In order to reduce re-encryption and file scanning log verbosity, when a configuration is imported or replicated to a cluster, PingFederate no longer scans files in the `etc` directory.

Other improvements

Info

- We also updated the following bundled components and third-party dependencies:
  - PingID Integration Kit 2.24
  - PingID Adapter 2.13.2
  - PingID PCV (with integrated RADIUS server) 3.0.3

PingFederate 11.1.5 (February 2023)

Resolved issues

Server log warnings

Fixed PF-33037

We've added a warning to server logs if the *ds-pwp-state-json* attribute is not present in PingDirectory's LDAP Response. This warning appears in the log every time a user interacts with the profile management page. Please enable this attribute to adhere to PingDirectory's security configuration best practices. PingDirectory version 8.1 and later supports this attribute, and customers running older versions are encouraged to upgrade to a supported version as soon as possible.

PingFederate 11.1.4 (February 2023)

Resolved issues

OAuth client management

Fixed PF-32790

When managing OAuth clients, we've resolved a defect where selecting the **Require JWT Secured Authorization Response Mode** text toggled the incorrect check box.

Potential security vulnerability

Fixed PF-32805

We've resolved a potential security vulnerability that is described in security advisory SECADV033.

Informing adapters of end policy result

Fixed PF-32890

When processing policy fragments, all adapters invoked in the fragment now correctly execute their respective post-processing step (if applicable) to inform the adapter of the end policy result.

Managing certificates within Metadata Export

Fixed PF-32965

Managing certificates within the **Metadata Export** flow no longer displays or saves an empty list of certificates, clearing out existing ones in the process. For more information, see Metadata export.

Cluster data replication

Fixed PF-32983

We've resolved a defect where cluster data replication could remove keys from engine node's `pf.jwk` file instead of merging and retaining the keys.

PingFederate 11.1.3 (December 2022)

Resolved issues

Improvements to custom revocation checker

Fixed PF-32395

We've improved PingFederate's custom revocation checker, ensuring that when the server returns stapled Online Certificate Status Protocol (OCSP) responses, PingFederate invokes the checker. Previously, PingFederate used the default revocation checker to validate these responses, which could cause single sign-on (SSO) failures with BCFIPS mode enabled. For more information, see Configuring certificate revocation.

Cluster replication notifications

Fixed PF-32398

We've improved notifications to signal to administrators that in the event of a replication failure or any changes to cluster configuration require replication. For more information, see Cluster management.

Null pointer exception during dependency error detection

Fixed PF-32553

During PingFederate dependency error detection, OGNL expressions in adapter-to-adapter mappings no longer raise a null pointer exception (NPE).

PingFederate updates to HSM ordering

Fixed PF-32556

We've updated the recommended security provider ordering for the Thales Luna Network hardware security module (HSM) to address an issue where temporary keys and sessions could accumulate on the HSM, eventually resulting in resource exhaustion. A limitation of the new ordering is that EC certificates can no longer operate as SSL server certificates. For details on the new order, see Integrating with Thales Luna Network HSM.

PingFederate 11.1.2 (October 2022)

Resolved issues

Bulk import for IdP connections

Fixed PF-31870

Resolved an issue where bulk import fails for identity provider (IdP) connections that fulfill Persistent Grant Extended Attributes.

Connection failures on external LDAP authentication login

Fixed PF-32001

PingFederate now recovers from initial connection failure when logging into the administrative console using external LDAP authentication.

Hiding user information from authentication API responses

Fixed PF-32028

You can now configure the setting `IncludeUserInfoInResponses` in the `<install dir>/server/default/data/config-store/org.sourceid.saml20.domain.mgmt.impl.AuthnApiManagerImpl.xml` file to hide user information from authentication API responses.

Errors on policy fragments configured to handle failures locally

Fixed PF-32073

When an error occurs on policies containing fragments and configured to handle failures locally, PingFederate no longer redirects a user to the service provider (SP) error page on SP-initiated single sign-on (SSO).

Outbound TLS connection failures

Fixed PF-32199

The certificate path-building algorithm now uses PingFederate's custom revocation checker. This fix resolves a bug where outbound TLS connections failed for servers that presented out-of-order certificate chains.

PingDirectory user registration

Fixed PF-32241

During user registration, PingFederate now sends all passwords to PingDirectory, resolving an issue where passwords consisting of only spaces would not properly register a PingDirectory password.

Configurations with no connection type in Kerberos realm

Fixed PF-32274

When reading the `pingfederate-kerberos-realms.xml` file, PingFederate no longer raises an error for configurations with no connection type in the Kerberos realm.

PingFederate 11.1.1 (July 2022)

Resolved issues

Security around password expiration

Fixed PF-29706 PingDirectory

Improved the security around password expiration when using PingDirectory as a user store.

Issuance criteria in authentication policy contracts

Fixed PF-31485

Issuance criteria in authentication policy contracts no longer cause the logs to indicate invalid XML errors. This issue did not cause runtime errors.

HTTP header for client IP addresses

Fixed PF-31735

Resolved an issue that sometimes occurred when IPV6 addresses were specified in the **HTTP Header for Client IP Addresses** field on the **Incoming Proxy Settings** window.

Error descriptions

Fixed PF-31753

PingFederate error descriptions no longer disclose details of java classes.

MasterKeyEncryptor failure during cluster replication

Fixed PF-31795

When PingFederate is using a custom MasterKeyEncryptor that relies on an SSL call to an external service, cluster replication no longer causes cascading failures because PingFederate is unable to open Java key store files.

Updating the client secret with the OAuth client management service

Fixed PF-31851

When updating the client secret with the OAuth client management service, PingFederate now correctly creates the secondary secrets.

OAuth authorization requests with `response_mode=pi.flow`

Fixed PF-31942

Now when PingFederate receives an OAuth authorization request with `response_mode=pi.flow`, password change and account recovery flows using an authentication policy work correctly.

Administrative API enhancement

Info

Improved the administrative API to manage the System for Cross-domain Identity Management (SCIM) inbound provisioning settings in identity provider (IdP) connections.

Message customization enhancement

Info

Enhanced PingFederate message customization by adding the following FedHub-specific context variables:

- `FedHubSpConnApplicationName`
- `FedHubSpConnName`
- `FedHubOAuthClientId`
- `FedHubOAuthClientName`

Cluster management enhancement

Info

Revised the **Cluster Management** window to make it more obvious when changes to the configuration on the administrative node have not been replicated to the engine nodes.

PingFederate 11.1 (June 2022)

New features and improvements in PingFederate 11.1.

New features and enhancements

PingOne integration

New PingOne

We've added Kerberos authentication via PingOne and the PingOne LDAP Gateway Data Store. This new capability allows PingFederate in the cloud, without a direct connection to Active Directory, to complete Kerberos authentication for browser-based SSO requests and STS transactions through PingOne.

JWT Secured Authorization Response Mode (JARM)

New

We’re proud to support JWT Secured Authorization Response Mode (JARM) in version 11.1. JARM allows authorization servers to transmit authorization responses in JSON web tokens (JWTs), providing digital signature and encryption, sender authentication, and audience restriction. As JARM becomes a requirement in FAPI 2, you can deploy open banking solutions confidently.

JWT Response for OAuth Token Introspection

New

We’re also introducing support for JWT Response for OAuth Token Introspection, a draft specification on track to become one of the authorization server requirements in the FAPI 2 Advanced Profile. JWT-secured introspection responses provide stronger assurance to the introspection requesters, most relevant when the requester, such as a resource server, expects to receive verified claims from the authorization server.

Client secret management

New

Seamless client secret rotation no longer requires real-time coordination between PingFederate administrators and the application development teams. You can now configure PingFederate to retain previous secrets for a configurable period, during which the application teams can work on updating the client secrets in their apps. This enhancement drastically lowers the costs of securing applications that use client secrets for authentication. For more information, see "Client Secret Retention Period" in the topic Managing client configuration defaults.

API support for Device Authorization Grant

New

In addition to template-driven user experience, the user authorization step from Device Authorization Grant supports API now. You can also decide whether PingFederate should check the device activation code before or after authentication. These new capabilities enable you to build applications with the desired user experience for input-constrained devices, such as smart TVs or telepresence equipment.

Amazon DynamoDB for grants

New

You can store OAuth persistent grants in Amazon DynamoDB, which allows you to take advantage of a NoSQL database where it matters most: delivering responsive experiences to globally distributed users and offering high availability at ease.

Revocation of self-contained access tokens

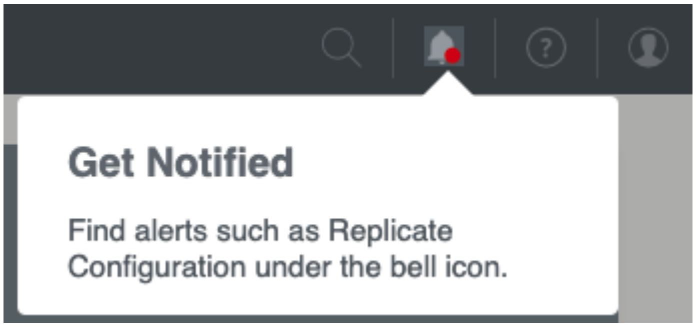
New

You can optionally enable direct revocation for self-contained access tokens (JWT access tokens). This flexibility provides a secure way to invalidate access tokens without revoking the underlying refresh tokens or persistent grants. For more information, see Configuring an access token management instance and its description of the **Enable Token Revocation** check box.

A new alert system

New

PingFederate 11.1 centralizes alerts, such as the reminder to replicate configuration, under the new **bell** icon in the top menu. You can review important alerts from any configuration window.



Get Notified

Find alerts such as Replicate Configuration under the bell icon.

Copy-and-paste authentication policies and fragments

New

Previously, if you wanted to update an authentication policy or a reusable policy fragment midstream, they had to reconfigure all downstream paths, which can take some effort. With PingFederate 11.1, you can copy a subtree of policy paths before removing a step (such as an IdP adapter), adding a new step (such as a selector or another IdP adapter), and then pasting the subtree back to the policy. This new capability applies to reusable policy fragments and between authentication policies and reusable policy fragments.

Administrative API to move individual policies

New

You can use the administrative API to move an individual policy to a specific location. This enhancement makes re-organizing policies by API requests easier and safer.

Cluster configuration management

New

PingFederate engine nodes now capture common configuration replication issues in their server logs and send replication status back to the console node. The **Cluster Management** window provides live updates when you select **Replicate Configuration** in the **Cluster Management** window. If an error occurs, you can act on it immediately and recover from potential outages faster.

Passthrough IdP Adapter

New

You can now associate authentication sessions with user identities passed through the new Passthrough Identity Provider (IdP) Adapter. By placing the Passthrough IdP Adapter downstream from an IdP connection in a policy tree, you can take advantage of additional capabilities associated with defining a user key. For example, you can use the user key to query or revoke a user's authentication sessions.

Kerberos authentication and ObjectSID

New

The Kerberos Adapter and the Kerberos Token Processor now return the `ObjectSID` attribute value. Because `ObjectSID` uniquely identifies the user in Active Directory, leveraging it helps streamline the **Attribute Source & Lookup** configuration.

Kerberos authentication and re-authentication

New

You can configure the Kerberos Adapter to fail when the service provider asks for re-authentication by including `ForceAuthn=true` (SAML 2.0) or `prompt=login` (OpenID Connect) in their authentication requests. For example, suppose user interactions are required when the partners ask for re-authentication. In that case, you can add the HTML Form Adapter to the **Fail** policy path of the Kerberos Adapter.

More error handling options

New

- You now can configure individual authentication policies to handle authentication failures locally without redirecting to the service providers or returning error messages to the OAuth clients. This flexibility addresses the scenario where an IdP-oriented end-user experience is desirable.
- PingFederate now includes error results from issuance criteria in error responses. Partners can use the error results to resolve issues as needed. If the invoked policy is configured to handle failures locally, you can do the same to improve the end-user experience.
- You can now optionally configure the HTML Form Adapter not to return control to PingFederate when an account lockout occurs. Instead, PingFederate returns a “please try again later” message to the browser or the authentication API application.

Extended properties for end-user interactions

New

You can now leverage extended properties in Velocity templates when customizing template-driven end-user interactions. You can reference extended properties in the templates instead of creating multiple `If` / `ElseIf` / `Else` directives, significantly reducing the initial effort. New and updated experiences can be inherited from extended property values from the OAuth client records and Browser SSO connections, eliminating most of the maintenance costs. PingFederate also passes extended property values to authentication API applications. As a result, application developers who create and maintain end-user UX for customer identities will benefit from this new enhancement.

Better documentation in Velocity templates

New

We’ve also improved inline documentation in our Velocity templates. Moving forward, we will maintain variable names and their definitions consistently to communicate changes, such as introducing new variables.

Enhancements in Thales HSM integration

New

Both Java 11 and 8 environments are supported when integrating with Thales Luna Cloud Hardware Security Module (HSM) Services or Luna Network HSMs. For more information about Thales Luna HSM Client, see the Luna Cloud HSM Service Client Guide and Luna Network HSM Documentation Archive.

Secondary signing certificate

New

You can now add a secondary signing certificate to your connections. If configured, PingFederate includes it in both the metadata exports and the metadata URL responses. This flexibility allows you to notify your partners about upcoming changes more easily through metadata.

Administrative API improvements

New

We improved the PingFederate administrative API to manage the following configurations:

- JIT provisioning settings in IdP connections
- **System > Data & Credential Stores > Identity Store Provisioners**
- **System > Server > General Settings**
- **System > Server > WS-Trust Settings**

Other improvements

New

- We significantly improved our metrics exposed through HTTP (at the heartbeat endpoint) and JMX to help you detect and diagnose performance issues. Both channels include HTTP response code counts, data source response time statistics, and Jetty queue size information; these metrics help troubleshoot latency issues associated with datastores or traffic volume.
- PingFederate now uses OCSP to obtain certificate revocation status by default on new installations. As part of this enhancement, PingFederate uses the OCSP responder URL provided in the certificate first, followed by the now optional Default OCSP Responder URL, and lastly, CRL, making the certificate validation process more efficient.
- The administrative console now provides guidance when you attempt to import a configuration archive obtained from a different version of PingFederate.
- PingFederate 11.1 supports Amazon IAM roles for service accounts, which increases security posture with credential isolation and auditability.
- PingOne Verify is now part of the PingFederate distribution `.zip` file and Windows installer.
- We also updated the following bundled components and third-party dependencies:
  - PingID Integration Kit 2.17
  - PingOne Fraud Integration Kit 1.0
  - PingOne Protect Integration Kit 1.2

  - Jackson-Databind 2.12.7
  - Log4j2 2.17.2
  - Spring Framework 5.3.20

Resolved Issues

H2 database engine upgrade

Fixed PF-21198

Upgraded the H2 database engine to version 2.1.210.

A username in the URL during change password flows

Fixed PF-24501

The username no longer appears in the URL during change password flows.

Guava upgrade

Fixed PF-28932

Upgraded the Guava dependency to version 30.1.1.

OAuth client Issuer DN

Fixed PF-29368

If the administrative API was used to create an OAuth client that has the Client Certificate authentication type, and the client's Issuer DN does not have a normalized DN value, the administrative console's **Client** window no longer fails to show the Issuer DN as the default value. This issue didn't affect runtime behavior.

Time stamp for last update

Fixed PF-29761

When a user record in a datastore mistakenly has a future date for the last update time, PingFederate no longer uses that date as the value of `attrib_last_timestamp` in the `channel_variable` table. Instead, PingFederate sets the value to the maximum time stamp that is not in the future.

Number and Boolean data types in JSON responses from REST API data source lookups

Fixed PF-29835

The JSON response from REST API data source lookups now retains number and Boolean data types instead of converting them to strings.

`NotYetConnectedException` warning messages from JGroup in the `server.log`

Fixed PF-30075

Resolved an issue that caused the `NotYetConnectedException` warning message to repeatedly appear in the `server.log` when using AWS_PING for dynamic cluster discovery.

Matching OAuth client's redirection URIs

Fixed PF-30146

If the OAuth client's redirection URI contains a wild card in the authority part of the URI, and the `redirect_uri` parameter of the token request contains userinfo in the authority part, then PingFederate will no longer consider the redirection URI a match.

Potential security vulnerability

Fixed PF-30255

Resolved a potential security vulnerability.

Logging invalid assertion errors

Fixed PF-30495

In a specific case, when PingFederate logs an invalid assertion error, the error message no longer fails to include a remark about why the assertion or response is invalid.

Null pointer exception in authentication API password reset flow

Fixed PF-30558

When an OAuth client is performing a password reset through the authentication API, if PingFederate does not find any session attributes, now PingFederate logs an error state instead of a null pointer exception.

Determining authentication instants for flows

Fixed PF-30770

Resolved an issue that prevented PingFederate from correctly determining the authentication instant for the flow when the initial OIDC authorization request specifies a max_age, the flow falls through to legacy authentication source selection (policies are disabled or no policy applies), and the user chooses an upstream OIDC IdP connection.

Templates for PingOne MFA 1.6.1

Fixed PF-30806 PingOne MFA

PingFederate now includes all the templates for PingOne MFA 1.6.1.

Dependency errors for SAML token processors and generators

Fixed PF-31054

When saving SAML token processors or generators, PingFederate now correctly handles dependency errors caused by misconfigured settings on the **Protocol Settings** window's **Federation Info** tab.

Preserving the order of map type configurations

Fixed PF-31145

Now PingFederate preserves the order of map type configurations under `<pf_install>/pingfederate/server/default/data/config-store` when performing a bulk export or a GET operation at the `/configStore` administrative API endpoint.

Warning about using the administrative console in multiple tabs

Fixed PF-31280

Now if you use the PingFederate administrative console in multiple tabs on one browser, it warns you that doing so might cause inconsistent behavior which could corrupt its configuration.

Saving authorization server settings overwrites scope.whitelist

Fixed PF-31304

Resolved an issue that caused PingFederate to overwrite the `scope.whitelist` in the `\data\config-store\org.sourceid.oauth20.domain.AuthzServerManagerImpl.xml` file when you save the authorization server settings.

OAuth client IDs added to admin.log entries

Fixed PF-31561

Now OAuth client MODIFY, CREATE, and DELETE event log entries in the `admin.log` include the client ID.

Honoring the property for maximum HTTP request body size

Fixed PF-31575

Now PingFederate honors the value of `http.maxRequestBodySize` in the `run.properties` file, which specifies the maximum HTTP request body size of any incoming request to PingFederate's web services and administrative API.

Known issues and limitations

Administrative console and administrative API

Issue

- /bulk: Only resource types currently supported by the administrative API are included in the exported data. We don't intend to introduce administrative API support to the following areas:
  - SAML 2.0 IdP Discovery
  - SAML 2.0 SP Affiliation
  - SMS Provider

- Previously, the administrative API did not accurately reflect a **Persistent Grant Max Lifetime** setting of 29 days (or shorter) with the selection of the **Grants Do Not Timeout Due To Inactivity** option. As a result, if you have configured such OAuth authorization server settings and have generated a bulk export in version 10.0 through 10.0.2, we recommend that you re-generate a new bulk export after upgrading to version 10.0.3 (or a more recent version). The newly exported data does not contain the aforementioned flaw, and you can safely import it to version 10.0.3 (or a more recent version).
- When enabling mutual TLS certificate-based authentication, administrators often configure a list of acceptable client certificate issuers. When you use a browser to access the console or the administrative API documentation, PingFederate returns to the browser the list of acceptable issuers as part of the TLS handshake. If the browser's client certificate store contains multiple client certificates, the browser often presents you only the certificates whose issuer matches one of the acceptable issuers. However, when PingFederate runs in a Java 11 environment, Chrome presents you all its configured client certificates, regardless of whether the issuer matches one of the acceptable issuers or not.
- Prior to toggling the status of a connection with the administrative API, you must ensure that any expired certificates or no longer available attributes are replaced with valid certificates or attributes; otherwise, the update request fails.
- When creating or updating a child instance of a hierarchical plugin, the administrative API retains objects with an `"inherited": false` name/value pair (or without such name/value pair altogether), ignores those with a value of `true`, and returns a 200 HTTP status code. No error messages are returned for the ignored objects.
- Using the browser's navigation mechanisms (for example, the **Back** button) causes inconsistent behavior in the administrative console. Use the navigation buttons provided at the bottom of windows in the PingFederate console.
- Using the PingFederate console in multiple tabs on one browser might cause inconsistent behavior which could corrupt its configuration.
- If authenticated to the PingFederate administrative console using certificate authentication, a session that has timed out might not appear to behave as expected. Normally (when using password authentication), when a session has timed out and a user attempts some action in the console, the browser is redirected to the sign-on page, and then back to the administrative console after authentication is complete. Similar behavior applies for certificate authentication, in principle. However, because the browser might automatically resubmit the certificate for authentication, the browser might redirect to the administrative console and not the sign on page.

TLSv1.3

Issue

For Java versions that don't support TLSv1.3 (meaning versions earlier than 8u261), PingFederate fails on start up with a `NoSuchAlgorithmException` exception. To resolve this error, remove `TLSv1.3` from the following settings in the `run.properties` file:

- `pf.tls.client.protocols`
- `pf.tls.runtime.server.protocols`
- `pf.tls.admin.server.protocols`

TLS cipher suite customization

Issue

PingFederate's TLS cipher suites can be customized by modifying `com.pingidentity.crypto.SunJCEManager.xml` (or a similarly-named file if BCFIPS or a hardware security module (HSM) is configured). After updating the file and replicating, all cluster nodes must be restarted for the change to take effect.

Java

Issue

- As of PingFederate 11.1, BC-FIPS and HSMs are not supported when using Java 17.
- Updating Java version 8 to version 11 results in an error when PingFederate is already installed and running. To work around this issue, uninstall and reinstall the PingFederate Windows service by running the `UninstallPingFederateService.bat` and `InstallPingFederateService.bat` files located in `<pf_install>/pingfederate/sbin/wrapper`.

Hardware security modules (HSM)

Issue

- For Entrust HSMs, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- For Entrust HSMs, PingFederate must be deployed with Oracle Server JRE 8 or Amazon Corretto 8.
- For keys stored in AWS CloudHSMs, JWT token signing fails when using RSASSA-PSS SHA-512.
- For keys stored in Thales HSMs, JWT token decryption fails when using RSAES OAEP with AES-CBC-192 or AES-CBC-256. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys and is consuming a token encrypted with one of these keys.
- When PingFederate is configured in hybrid mode with a Thales HSM, it is not possible to export a locally-stored EC key pair.
- When PingFederate is configured in hybrid mode with a Thales HSM, JWT token decryption using ECDH-ES may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored locally, and PingFederate is consuming a token encrypted with this key.
- TLS 1.3 is not currently supported with any HSM.

SSO and SLO

Issue

- When consuming SAML metadata, PingFederate does not report an error when neither the `validUntil` nor the `cacheDuration` attribute is included in the metadata. Note that PingFederate does reject expired SAML metadata as indicated by the `validUntil` attribute value, if it is provided.
- The anchored-certificate trust model cannot be used with the Single log off (SLO) redirect binding because the certificate cannot be included with the logout request.
- If an IdP connection is configured for multiple virtual server IDs, PingFederate will always use the default virtual server ID for IdP Discovery during an SP-initiated SSO event.

Composite Adapter configuration

Issue

SLO is not supported when users are authenticated through a Composite Adapter instance that contains another instance of the Composite Adapter.

Self-service password reset

Issue

Passwords can be reset for Microsoft Active Directory user accounts without the permission to change password.

OAuth

Issue

PingFederate does not support a case-sensitive naming convention for OAuth client ID values when client records are stored in a directory server. For example, after creating a client with an ID value of `sampleClient`, PingFederate does not allow the creation of another client with an ID value of `SampleClient`.

Although it's possible to create clients using the same ID values with different casings when client records are stored in XML files, a database server, or custom storage, we recommend not doing so to avoid potential record migration issues.

Customer identity and access management

Issue

Some browsers display a date-picker user interface for fields that have been designed for date-specific inputs. Some browsers do not. If one or more date-specific fields are defined on the registration page or the profile management page (or both), end users must enter the dates manually if their browsers do not display a date-picker user interface for those fields.

Provisioning

Issue

- LDAP referrals return an error and cause provisioning to fail if the `user` or `group` objects are defined at the DC level, and not within an OU or within the Users CN.
- The `totalResults` value in SCIM responses indicates the number of results returned in the current response, not the total number of estimated results on the LDAP server.

Logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

Database logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

RADIUS NAS-IP-Address

Issue

The RADIUS NAS-IP-Address is only included in Access-Request packets when the `pf.bind.engine.address` is set with an IPv4 address. IPv6 is not supported.

Deprecated features

Microsoft Internet Explorer 11

Info

Ping Identity commits to deliver the best experience for administrators and users. As we continue to improve our products, we encourage you to migrate off of Microsoft Internet Explorer 11. Starting with PingFederate 11.0, Internet Explorer 11 is no longer included in the PingFederate qualification process for administrators or users. For a list of supported browsers, see System requirements.

Configcopy tool, Connection Management Service, SSO Directory Service

Info

As of PingFederate 10.2, these features have been deprecated and will be removed in a future release.

Oracle Directory Server Enterprise Edition

Info

As Oracle ended its Premier Support for Oracle Directory Server Enterprise Edition (ODSEE 11g) in December 2019, we no longer include ODSEE as part of the PingFederate qualification process (starting with PingFederate 10.2). We continue to qualify against Oracle Unified Directory (www.oracle.com/middleware/technologies/unified-directory.html) and other supported directory servers. For a full list, see System requirements.

SNMP

Info

Starting with PingFederate 10.2, monitoring and reporting through the Simple Network Management Protocol (SNMP) has been removed.

Roles and protocols

Info

Starting with PingFederate 10.1, roles and protocols are always enabled and no longer configurable through the administrative console and API.

S3_PING discovery protocol

Info

Starting with PingFederate 10.1, the S3_PING discovery protocol has been deprecated. Customers running on AWS infrastructure should instead use NATIVE_S3_PING.

Red Hat Enterprise Linux install script

Info

Starting with PingFederate 10.0, the Red Hat Enterprise Linux install script is no longer available. To install PingFederate 10.0 for Linux, you must download and extract the product distribution `.zip` file.

PingFederate 11.0.10 - April 2024

Resolved issues

Rest datastore security vulnerability

Security PF-34720

Fixed a JSON injection vulnerability in REST datastores described in security advisory SECADV044.

Runtime nodes security vulnerability

Security PF-34896

Fixed a path traversal vulnerability in Runtime nodes described in security advisory SECADV044.

OpenID Connect policy management editor security vulnerability

Security PF-35081

Fixed a Cross-Site Scripting vulnerability in the OpenID Connect Policy Management Editor described in security advisory SECADV044.

Slow log consumption affects performance

Fixed PF-33368

Fixed a defect that caused performance issues for PingFederate when third-party logging services were slow to consume logging events.

PingFederate 11.0.9 (December 2023)

Resolved issues

Fixed PF-29706

Fixed a Server-Side Request Forgery vulnerability in the Initial Setup Wizard described in security advisory SECADV041.

PingFederate 11.0.8 (August 2023)

Resolved issues

Logging validation

Fixed PF-34017

We've improved logging validation.

Potential security vulnerability

Fixed PF-33449

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

Potential security vulnerability

Fixed PF-34017

We've resolved a potential security vulnerability that is described in security advisory SECADV037.

PingFederate 11.0.7 (February 2023)

Resolved issues

Server log warnings

Fixed PF-33037

We've added a warning to server logs if the *ds-pwp-state-json* attribute is not present in PingDirectory's LDAP Response. This warning appears in the log every time a user interacts with the profile management page. Please enable this attribute to adhere to PingDirectory's security configuration best practices. PingDirectory version 8.1 and later supports this attribute, and customers running older versions are encouraged to upgrade to a supported version as soon as possible.

PingFederate 11.0.6 (February 2023)

Resolved issues

Potential security vulnerability

Fixed PF-32805

We've resolved a potential security vulnerability that is described in security advisory SECADV033.

PingFederate 11.0.5 (October 2022)

Resolved issues

IPV6 address issue

Fixed PF-31735

Resolved an issue that sometimes occurred when IPV6 addresses were specified in the **HTTP Header for Client IP Addresses** field on the **Incoming Proxy Settings** window.

Administrative console login

Fixed PF-32001

PingFederate now recovers from initial connection failure when logging into the administrative console using external LDAP authentication.

User registration defect resolution

Fixed PF-32241

During user registration, PingFederate now sends all passwords to PingDirectory, resolving an issue where passwords consisting of only spaces would not properly register a PingDirectory password.

PingFederate 11.0.4 (August 2022)

Resolved issues

MasterKeyEncryptor and cluster replication

Fixed PF-31795

When PingFederate uses a custom MasterKeyEncryptor that relies on an SSL call to an external service, cluster replication no longer causes cascading failures because PingFederate cannot open Java key store files.

Rule matching for fragment nodes and NullPointerException

Fixed PF-31929

When using rule matching for fragment nodes, PingFederate no longer raises a NullPointerException (NPE) if a fragment fails.

Zero byte archives

Fixed PF-31966

Resolved an issue that caused PingFederate to generate a zero byte archive when it couldn’t read a file in the `<pf_install>/pingfederate/server/default/data` directory.

JWT access token lifetimes

Fixed PF-31989

When using centralized and dynamically rotating keys for OAuth and OpenID Connect, PingFederate now prevents you from setting the JWT access token lifetime to be longer than the `dynamic-rotation-period-in-days` specified in `<pf_install>/pingfederate/server/default/data/config-store/jwks-endpoint-configuration.xml`.

PingFederate 11.0.3 (May 2022)

Resolved issues

Intermittent failure to respond after restart caused by LDAP SDK

Fixed PF30776

To resolve an issue in which PingFederate occasionally stopped responding after a restart, the UnboundID LDAP SDK for Java was updated to version 6.0.4.

TLS 1.3 for outbound connections

Fixed PF-31303

PingFederate now supports TLS 1.3 for outbound connections when running on Java 8 versions 8u261 and newer.

Updated Spring Framework

Info PF-31169

Updated Spring Framework to version 5.3.18.

PingFederate 11.0.2 (March 2022)

New features and enhancements

Updated PingOne MFA adapter

Info PingOne MFA

Updated the bundled PingOne MFA adapter to version 1.6.

Resolved issues

LDAP connections

Fixed PF-30804

Resolved an issue that caused LDAP connections to periodically fail during provisioning.

Bulk export

Fixed PF-30863

Bulk export no longer fails to include all XML OAuth clients in the response payload.

Single sign-on from browsers on iOS

Fixed PF-31057

Resolved an issue that caused single sign-on from browsers on iOS to fail when an authentication policy terminates on Kerberos Adapter fallback that has an existing session.

nCipher mode

Fixed PF-31064

When running PingFederate in nCipher mode, now the administrative API successfully generates elliptic curve (EC) keys when the optional signatureAlgorithm field is not provided.

TLS 1.3 for inbound connections

Fixed PF-31112

PingFederate now supports TLS 1.3 for inbound connections when running on Java 8 versions 8u261 and newer.

Symantec VIP Adapter

Fixed PF-31123

Resolved an issue that prevented PingFederate from using the Symantec VIP Adapter.

LDAP-related performance

Fixed PF-31146

Resolved an LDAP-related performance issue.

Signature verification for certificate revocation lists

Fixed PF-31159

Resolved an issue where signature verification for certificate revocation lists could take more than 10 seconds on Windows. When LDAP-based authentication was enabled in the administrative console, this could prevent administrative users from signing on.

PingFederate 11.0.1 (January 2022)

New features and enhancements

Rolling grace period for refresh tokens

Improved

When PingFederate rotates a refresh token, if the client fails to get the new token, now PingFederate can accept the previous token for the short period that you specify with the **Refresh Token Rolling Grace Period** setting.

Performance improvement

Info

Improved performance of the administrative console when a large number of OAuth clients are stored in LDAP or JDBC datastores.

URL region of the PingOne home button

Info PingOne

When configuring the URL of the PingOne home button in the PingFederate administrative console, now `pf.pingone.admin.url.region` in `run.properties` supports `Canada` as a region.

AWS CloudHSM client

Info

PingFederate can be successfully integrated with AWS CloudHSM client version 3.4.4.

Resolved issues

Resolved a potential security vulnerability

Security PF-30450

Resolved a potential security vulnerability that is described in security bulletin SECBL021.

Updated Apache Log4j2

Security PF-30536

Resolved a potential security vulnerability by updating Apache Log4j2 to version 2.17.1.

Authenticating PingDirectory users

Fixed PF-30557 PingDirectory

Resolved an issue that allowed PingDirectory users to authenticate with expired passwords.

Certificate revocation list checks

Fixed PF-30637

Resolved an issue that caused certificate revocation list (CRL) checks to return "`issuer not found in trusted CAs store`" even though the issuer certificate is present.

PingFederate 11.0 (December 2021)

New features and improvements in PingFederate 11.0.

New features and enhancements

PingOne LDAP Gateway datastore

New PingOne

PingFederate in the cloud can now connect to on-premise directory servers through the PingOne LDAP Gateway. This new capability reduces the complexity of moving to the cloud, while maintaining connectivity to on-premise end-user data.

PingOne unified admin integration

New PingOne

Administrators can now open the PingOne unified admin from any configuration window in the PingFederate administrative console. To activate the new Home icon, enter the PingOne region and the environment ID in the `run.properties` file.

Management of configuration encryption keys

New

PingFederate maintains a set of configuration encryption keys to encrypt sensitive configuration information provided by the administrators and decrypt them later as needed. While we continue recommending customers to protect their configuration encryption keys by AWS KMS or custom solutions based on the PingFederate SDK (the `MasterKeyEncryptor` interface), we are introducing two enhancements in this area.

- Key rotatation: Administrators or key-management processes can now insert a new configuration encryption key into the system with one click in the administrative console or a single administrative API request. Once rotated, PingFederate starts using this new encryption key when it needs to encrypt sensitive configuration data.
- Re-encryption of configuration data: Version 11 also comes with a new `configkeymgr` command-line utility. Administrators can optionally scan, review, re-encrypt, and delete older configuration encryption keys in their systems. Furthermore, administrators can now choose to re-encrypt sensitive information when importing an archive from a different environment; this is most useful when administrators do not want to share configuration encryption keys between the two environments.

Secret Managers

New

The new Secret Managers support allows customers to store certain credentials, such as data store credentials, in external secret management systems and have PingFederate retrieve them as needed. It helps customers comply with internal IT policies or meet and exceed their industry standards. Version 11 integrates out-of-the-box with CyberArk Credential Provider. Customers can also develop custom solutions based on the PingFederate SDK (the `SecretManager` interface), to connect to other secret management systems.

FAPI 1 Advanced Final certifications

New

Ping Identity remains a solid contributor to the financial-grade API initiatives from the OpenID Foundation. We're proud that PingFederate is a certified implementation of various FAPI 1 Advanced Final profiles, including all profiles under Australia CDR and UK Open Banking and four profiles under Brazil Open Banking. Deploy Open Banking solutions with confidence and rest assured that we will continue to invest in OAuth, OpenID Connect, and FAPI specifications. For more information about OpenID certifications, visit https://openid.net/certification/#FAPI_OPs.

Flexibility in ID token issuance

New

When processing an OpenID Connect hybrid flow, in addition to issuing an ID token from the token endpoint, PingFederate may also return an ID token from the authorization endpoint, depending on the requested response type. Administrators now have the flexibility to separate these two ID token issuances and configure their fulfillment differently. These enhancements allow our customers to comply with the regulatory requirements and open standards set by the Australian CDR and FAPI specifications.

Encrypted request objects

New

PingFederate now supports encrypted request objects that OAuth clients send to its Authorization endpoint and the Pushed Authorization Requests (PAR) endpoint. As needed, administrators can make encrypted request objects mandatory. This new capability further secures the confidentiality of authentication request parameters.

Authorization server issuer identification

New

The OAuth 2.0 Authorization Server Issuer Identification draft specification intends to mitigate the scenario where mix-up attacks are a potential threat to all OAuth clients interacting with multiple authorization servers. As needed, administrators can enable this optional capability.

Better private key JWT validation

New

In the context of OAuth client authentication, when processing private key JWTs from applications, PingFederate now ensures that the issuer (`iss`) claim value matches the client ID. This enhancement removes the need to use issuance criteria to enforce this validation requirement.

Message customization in OIDC IdP connection

New

PingFederate 11 can now take the request parameters from the SAML 2.0 SP or the OpenID Connect relying party (OIDC RP) into account when building its OIDC authentication request to the third-party OpenID Provider (OP). This capability allows administrators to selectively configure the values in the outbound OIDC authentication requests if their use cases or the third-party OPs have the need to gather more information from the originating SP or RP.

Multi-valued attribute format

New

Administrators can optionally indicate that PingFederate should always return an array for an attribute value regardless of whether the attribute contains one or multiple values. This flexibility simplifies the logic required to consume attribute values from access tokens or ID tokens.

Streamlined initial setup experience

New

We're pleased to introduce a brand new initial setup experience, where administrators can finish their initial setup in as little as four steps, rapidly making our rock-solid capabilities available after starting PingFederate for the first time.

Individual policy management by API

New

Administrators can now focus solely on one policy without including other policies as part of the API request when managing an individual authentication policy through the administrative API. This simplification improves the API experience and eliminates the risk of making unexpected changes in other authentication policies.

Console heartbeat

New

Monitoring the status of the console node is now more straightforward with the addition of the `/pf/heartbeat.ping` heartbeat endpoint to the administrative port. Like its runtime counterpart, the administrative heartbeat endpoint is also capable of returning additional information. If administrators want detailed information in the responses, set the `pf.heartbeat.system.monitoring` property to `true` in the `run.properties` file.

Datastore enhancements

New

- We expanded the REST API datastore with HTTP POST support. Administrators can connect to data repositories that prefer or require the HTTP POST method.
- Administrators can add attribute options in their LDAP directory searches. This enhancement expands what PingFederate can retrieve from the directory servers that support attribute options, PingDirectory being one of them.
- When configuring an LDAP search filter that uses one or more variables, an administrator can optionally specify default values for them, most useful in the scenarios where these variables may not contain any values at runtime.

Migration of templates

New

Our upgrade tools now copy customized default templates from the previous installation to the new one. This improvement preserves the end-user experience and branding, making it easier to verify and move forward with version 11 and beyond.

New configuration for dynamic discovery settings

New

Previously, administrators could only define dynamic discovery settings to discover cluster membership in the `server/default/conf/tcp.xml` file. Version 11 provides a new configuration file for these settings, `jgroups.properties` in the `bin` directory. This new approach streamlines future upgrade experiences. For new installations, we recommend defining dynamic discovery settings in the `jgroups.properties` file. While upgraded environments will continue to look for dynamic discovery settings from the `tcp.xml` file, we recommend performing a one-time migration to ease the upgrade experiences in the future.

Email ownership verification by OTP

New

For customer identities, in addition to email ownership verification by one-time link, administrators can now enable email ownership verification by one-time passcode (OTP). This new option offers a modern verification experience. It also helps customers who prefer not to send hyperlinks via email to their consumers.

Request context to authentication API applications

New

Administrators can optionally configure PingFederate to pass contextual information, such as the OAuth client ID or tracked HTTP parameters, from the sign-on requests to the authentication API applications. This allows developers to build applications that offer tailored experiences and satisfy branding requirements from their organizations based on contextual information from the sign-on requests.

Kerberos authentication improvement

New

Administrators can now ensure Kerberos authentication remains functional for service tickets associated with older Kerberos service account passwords after updating the **Domain/Realm Password** field with a new password in PingFederate. This optional capability increases productivity because workforce identities are no longer required to restart their Windows sessions in order to authenticate via Kerberos.

Contextual information in Session Management API responses

New

The Session Management API now includes IP address and User-Agent information in its responses. Clients with access to this API can learn more about their users and provide suitable offerings based on this new insight.

Security enhancements

New

- PingFederate now supports Amazon EC2 Instance Metadata Service version 2 (IMDSv2) when AWS_PING is the chosen dynamic discovery method. No PingFederate configuration changes are required, and IMDSv1 remains supported.
- PingFederate now records administrative timed-out events in the administrator audit log (`admin.log`).
- The **Change Password** and **Password Reset** end user-facing pages now time out after 30 minutes. This is the new default behavior for new and upgraded installations. As needed, administrators can configure a different **Password Update Timeout** value per HTML Form Adapter instance to suit the needs of their organizations.

Other improvements

New

- PingFederate now includes HTTP/2 support for inbound requests for better performance.
- Administrators can optionally configure PingFederate to mask values obtained from tracked parameters in the server log. Look for the `MaskTrackedParams` setting in the `org.sourceid.saml20.domain.mgmt.impl.TrackedHttpParamManagerImpl.xml` file.
- Administrators are free to enable the refresh token grant type independently on a per-client basis regardless of whether session validation is enabled in any Access Token Managers.
- Administrators can optionally configure PingFederate to redirect end-users back to the **Sign On** page after successfully updating their soon-to-expire password as part of their SSO requests.
- The **Reuse Existing Persistent Access Grants for Grant Types** authorization server setting is now overridable per client.
- PingFederate now supports RSAES OAEP using SHA-256 and MGF1 with SHA-256 (RSA-OAEP-256) when minting outbound ID tokens or processing inbound encrypted request objects
- Administrators can optionally restrict access to the redirectless mode per authentication API application. Additionally, administrators can further limit each application to an OAuth client to improve security around the redirectless mode of the authentication API.

- We upgraded the framework of our administrative API documentation to Swagger 2.0.
- PingFederate now preserves line breaks and indentations of OGNL expressions.
- The following templates now share the following Velocity template variables, which makes branding end-user experiences easier.

Templates	Variables
◦ `identifier.first.template.html` ◦ `html.form.login.challenge.template.html` ◦ `html.form.login.template.html` ◦ `html.form.message.template.html` ◦ `html.form.password.expiring.notification.template.html`	◦ `$client_id` - The ID of the OAuth client used by the request ◦ `$entityId` - The entity ID of the SP connection used by the request ◦ `$connectionName` - The name of the SP connection used by the request ◦ `$baseUrl` - The base URL of PingFederate instance ◦ `$adapterId` - The IdP adapter ID used by the request ◦ `$spAdapterId` - The SP adapter ID used by the request

- Updated the following bundled components and third-party dependencies:
  - Jetty 9.4.44
  - JGroups 4.2.16
  - jose4j 0.7.9
  - Log4j 2.16.0
  - PingFederate Agentless Integration Kit 2.0.4
  - PingID Integration Kit 2.15.0
  - PingOne Integration Kit 2.4.1
  - Spring Framework 5.3.5

Resolved issues

Cluster dynamic OAuth/OpenID Connect keys

Fixed PF-20709

Resolved an issue that sometimes caused a cluster's dynamic OAuth/OpenID Connect keys to fail to synchronize when a node restarts.

Provisioning

Fixed PF-27519

Resolved an issue that prevented a PingFederate provisioner from using a group of GUIDs as the source to detect new and removed records.

Configuring the `favicon.ico` URL

Fixed PF-28074

Now PingFederate correctly applies customizations of `response-header-runtime-config.xml` to the `favicon.ico` URL.

Retrieving OAuth clients from Oracle databases

Fixed PF-28842

Reduced the time it takes for PingFederate to retrieve OAuth clients from Oracle databases.

Unnecessary dependency error banners

Fixed PF-29189

Unnecessary dependency error banners no longer appear in the administrative console when you use the administrative API to modify selectors or service provider adapters.

Localizing end user messages from the authentication API

Fixed PF-29202

Now you can localize end user messages from the authentication API for registration failure scenarios.

Device authorization flow using IdP connection OAuth attribute mapping

Fixed PF-29294

Resolved an issue that stopped PingFederate from completing a device authorization flow when using IdP connection OAuth attribute mapping.

Multiple Sign-On Delay template redirects

Fixed PF-29318

When a proxy is in front of PingFederate, the Multiple Sign-On Delay template now redirects to the correct port.

Logging [.codeph]``XMLCipher::decryptElement called without a key and unable to

```
                            resolve``
[.ping_changetype-fixed]#Fixed#
[.ping_ticket]#PF-29352#
```

As a service provider (SP), when PingFederate can't decrypt an assertion using the primary encryption certificate, it now logs the following message at the WARN level instead of the ERROR level: "`XMLCipher::decryptElement called without a key and unable to resolve`".

Security vulnerability

Fixed PF-29381

Resolved a potential security vulnerability caused by web server URI mishandling.

Response headers for `/pf-ws` and `/pf-scim` endpoints

Fixed PF-29392

Introduced the ability to add response headers to the `/pf-ws` and `/pf-scim` endpoints.

Upgrade utility

Fixed PF-29470

Fixed the upgrade utility so that, in non-interactive mode, it retains cipher related settings that are different from the default settings in the source version. PingFederate changes to new default settings on upgrade only if the settings have not been changed from the defaults in the source install.

Custom template specified for the HTML Form Adapter

Fixed PF-29509

Resolved an issue that caused PingFederate to render the default `forgot-password-error.html` template instead of the custom template specified in the **Password Reset Error Template** field for the HTML Form Adapter.

Partial matches for resource URIs with OAuth 2.0 Token Exchange

Fixed PF-29668

Resolved an issue that prevented the use of partial matches for resource URIs with OAuth 2.0 Token Exchange and produced the error message: "`Unable to find a token generation policy instance to issue a token`".

Adding attributes to data source lookups

Fixed PF-29795

Now, when administrators add an attribute to a data source lookup but do not use the attribute anywhere, such as for contract mapping or issuance criteria, the attribute persists in the administrative console and API.

Microsoft Active Directory LDIF script for persistent grant storage

Fixed PF-29847

The Microsoft Active Directory LDIF script for persistent grant storage now creates an index for the `accessGrantGuid` attribute.

Notification publisher

Fixed PF-29870

Resolved the following notification publisher issues:

- When the SMTP server queues a message but has not sent it yet, the log now indicates that the message was queued, not that it was sent.
- PingFederate now respects the **Connection Timeout** setting for the notification publisher's SMTP server.
- Deprecated the **Retry Attempt** and **Retry Delay** fields for the notification publisher's SMTP server and removed them from the administrative console. PingFederate can still handle API configurations with those fields but they do nothing.

Target resources that don't start with `http://` or `https://`

Fixed PF-30002

Now target resources that don't start with `http://` or `https://` are also available for mapping and issuance criteria.

Response code for an invalid transport method

Fixed PF-30039

Now various endpoints return `400 Bad Request` instead of `500 Internal Server Error` when they receive requests with an invalid transport method. For example, calling the ACS endpoint with a `GET` instead of a `POST` now returns `400 Bad Request`.

Custom IDP adapters that use the class for filterable dropdown controls

Fixed PF-30232

The administrative console no longer shows an error message when you try to create an instance of a custom IDP adapter that uses the class for filterable dropdown controls, `ConnectionSelectionFieldDescriptor`.

Memory usage during certificate revocation list (CRL) parsing

Fixed PF-30272

Reduced memory usage during certificate revocation list (CRL) parsing, which speeds up CRL retrieval and avoids memory exhaustion in the case of very large CRLs.

Known issues and limitations

Administrative console and administrative API

Issue

- /sp/idpConnections: For identity provider (IdP) connections, the administrative API connection support is limited to Browser SSO, WS-Trust STS, and OAuth Assertion Grant connections. As a result, when updating an IdP connection using the administrative API, it is possible to lose inbound provisioning settings previously configured using the administrative console.
- /bulk: Only resource types currently supported by the administrative API are included in the exported data. Resources not yet supported include:
    - Identity Store Provisioners

  - Inbound provisioning settings from IdP connections
  - SMS Provider settings

- Previously, the administrative API did not accurately reflect a **Persistent Grant Max Lifetime** setting of 29 days (or shorter) with the selection of the **Grants Do Not Timeout Due To Inactivity** option. As a result, if you have configured such OAuth authorization server settings and have generated a bulk export in version 10.0 through 10.0.2, we recommend that you re-generate a new bulk export after upgrading to version 10.0.3 (or a more recent version). The newly exported data does not contain the aforementioned flaw, and you can safely import it to version 10.0.3 (or a more recent version).
- When enabling mutual TLS certificate-based authentication, administrators often configure a list of acceptable client certificate issuers. When an administrator uses a browser to access the console or the administrative API documentation, PingFederate returns to the browser the list of acceptable issuers as part of the TLS handshake. If the browser's client certificate store contains multiple client certificates, the browser often presents to the user only the certificates whose issuer matches one of the acceptable issuers. However, when PingFederate runs in a Java 11 environment, Chrome presents to the administrator all its configured client certificates, regardless of whether the issuer matches one of the acceptable issuers or not.
- Prior to toggling the status of a connection with the administrative API, an administrator must ensure that any expired certificates or no longer available attributes are replaced with valid certificates or attributes; otherwise, the update request fails.
- When creating or updating a child instance of a hierarchical plugin, the administrative API retains objects with an `"inherited": false` name/value pair (or without such name/value pair altogether), ignores those with a value of `true`, and returns a 200 HTTP status code. No error messages are returned for the ignored objects.
- Using the browser's navigation mechanisms (for example, the **Back** button) causes inconsistent behavior in the administrative console. Use the navigation buttons provided at the bottom of windows in the PingFederate console.
- Using the PingFederate console in multiple tabs on one browser might cause inconsistent behavior which could corrupt its configuration.
- If authenticated to the PingFederate administrative console using certificate authentication, a session that has timed out might not appear to behave as expected. Normally (when using password authentication), when a session has timed out and a user attempts some action in the console, the browser is redirected to the login page, and then back to the administrative console after authentication is complete. Similar behavior applies for certificate authentication, in principle. However, because the browser might automatically resubmit the certificate for authentication, the browser might redirect to the administrative console and not the login page.

TLSv1.3

Issue

For Java versions that don't support TLSv1.3 (meaning versions earlier than 8u261), PingFederate fails on start up with a `NoSuchAlgorithmException` exception. To resolve this error, remove `TLSv1.3` from the following settings in the `run.properties` file:

- `pf.tls.client.protocols`
- `pf.tls.runtime.server.protocols`
- `pf.tls.admin.server.protocols`

TLS cipher suite customization

Issue

PingFederate's TLS cipher suites can be customized by modifying `com.pingidentity.crypto.SunJCEManager.xml` (or a similarly-named file if BCFIPS or a hardware security module (HSM) is configured). After updating the file and replicating, all cluster nodes must be restarted for the change to take effect.

Updating Java 8 to Java 11

Issue

Updating Java version 8 to version 11 results in an error when PingFederate is already installed and running. To work around this issue, uninstall and reinstall the PingFederate Windows service by running the `UninstallPingFederateService.bat` and `InstallPingFederateService.bat` files located in `<pf_install>/pingfederate/sbin/wrapper`.

Hardware security modules (HSM)

Issue

- For Entrust HSMs or AWS CloudHSM, PingFederate must be deployed with Oracle Server JRE 8 or Amazon Corretto 8.
- For Entrust HSMs, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- For keys stored in Thales HSMs, JWT token decryption fails when using RSAES OAEP with AES-CBC-192 or AES-CBC-256. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys and is consuming a token encrypted with one of these keys.
- When PingFederate is configured in hybrid mode with a Thales HSM, it is not possible to export a locally-stored EC key pair.
- When PingFederate is configured in hybrid mode with a Thales HSM, JWT token decryption using ECDH-ES may fail. This issue only arises if PingFederate is configured with static OAuth and OpenID Connect keys, a static key is stored locally, and PingFederate is consuming a token encrypted with this key.
- TLS 1.3 is not currently supported with any HSM.

SSO and SLO

Issue

- When consuming SAML metadata, PingFederate does not report an error when neither the `validUntil` nor the `cacheDuration` attribute is included in the metadata. Note that PingFederate does reject expired SAML metadata as indicated by the `validUntil` attribute value, if it is provided.
- The anchored-certificate trust model cannot be used with the SLO redirect binding because the certificate cannot be included with the logout request.
- If an IdP connection is configured for multiple virtual server IDs, PingFederate will always use the default virtual server ID for IdP Discovery during an SP-initiated SSO event.

Composite Adapter configuration

Issue

SLO is not supported when users are authenticated through a Composite Adapter instance that contains another instance of the Composite Adapter.

Self-service password reset

Issue

Passwords can be reset for Microsoft Active Directory user accounts without the permission to change password.

OAuth

Issue

PingFederate does not support a case-sensitive naming convention for OAuth client ID values when client records are stored in a directory server. For example, after creating a client with an ID value of `sampleClient`, PingFederate does not allow the creation of another client with an ID value of `SampleClient`.

Although it's possible to create clients using the same ID values with different casings when client records are stored in XML files, a database server, or custom storage, we recommend not doing so to avoid potential record migration issues.

Customer identity and access management

Issue

Some browsers display a date-picker user interface for fields that have been designed for date-specific inputs. Some browsers do not. If one or more date-specific fields are defined on the registration page or the profile management page (or both), end users must enter the dates manually if their browsers do not display a date-picker user interface for those fields.

Provisioning

Issue

- LDAP referrals return an error and cause provisioning to fail if the `user` or `group` objects are defined at the DC level, and not within an OU or within the Users CN.
- The `totalResults` value in SCIM responses indicates the number of results returned in the current response, not the total number of estimated results on the LDAP server.

Logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

Database logging

Issue

- If a source attribute has been configured for masking in an IdP adapter or IdP connection and the source attribute is mapped to OAuth's persistent grant `USER_KEY` attribute, the `USER_KEY` attribute will not be masked in the server logs. Other persistent grant attributes will be masked.
- Even if a source attribute has been configured for masking in an IdP adapter and the source attribute is mapped as the adapter's unique user key, the user key attribute is not masked in the server or audit logs.

RADIUS NAS-IP-Address

Issue

The RADIUS NAS-IP-Address is only included in Access-Request packets when the `pf.bind.engine.address` is set with an IPv4 address. IPv6 is not supported.

Deprecated features

Microsoft Internet Explorer 11

Info

Ping Identity commits to deliver the best experience for administrators and users. As we continue to improve our products, we encourage our customers to migrate off of Microsoft Internet Explorer 11. Starting with PingFederate 11.0, Internet Explorer 11 is no longer included in the PingFederate qualification process for administrators or users. For a list of supported browsers, see System requirements.

Configcopy tool, Connection Management Service, SSO Directory Service

Info

As of PingFederate 10.2, these features have been deprecated and will be removed in a future release.

Oracle Directory Server Enterprise Edition

Info

As Oracle ended its Premier Support for Oracle Directory Server Enterprise Edition (ODSEE 11g) in December 2019, we no longer include ODSEE as part of the PingFederate qualification process (starting with PingFederate 10.2). We continue to qualify against Oracle Unified Directory (www.oracle.com/middleware/technologies/unified-directory.html) and other supported directory servers. For a full list, see System requirements.

SNMP

Info

Starting with PingFederate 10.2, monitoring and reporting through the Simple Network Management Protocol (SNMP) has been removed.

Roles and protocols

Info

Starting with PingFederate 10.1, roles and protocols are always enabled and no longer configurable through the administrative console and API.

S3_PING discovery protocol

Info

Starting with PingFederate 10.1, the S3_PING discovery protocol has been deprecated. Customers running on AWS infrastructure should instead use NATIVE_S3_PING.

Red Hat Enterprise Linux install script

Info

Starting with PingFederate 10.0, the Red Hat Enterprise Linux install script is no longer available. To install PingFederate 10.0 for Linux, you must download and extract the product distribution `.zip` file.

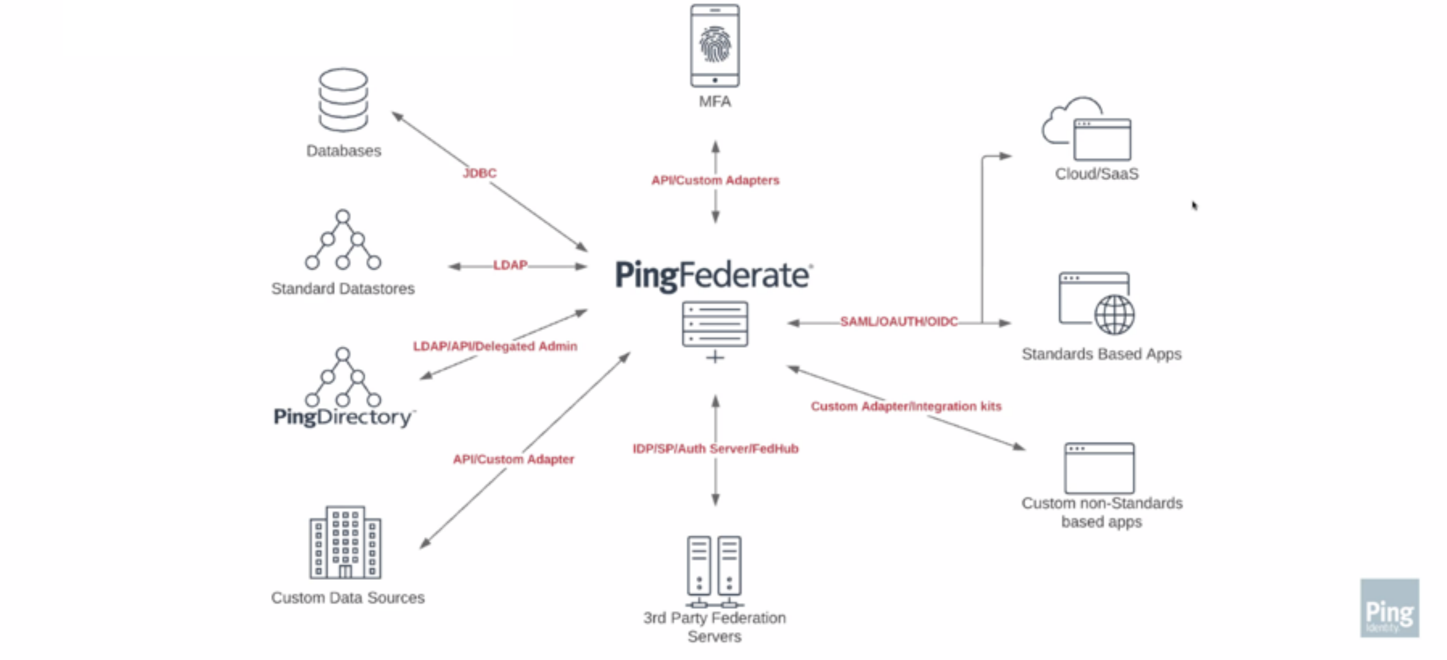
Previous releases

Find information about enhancements and issues resolved in previous releases of PingFederate in the Ping Documentation Archive.

Introduction to PingFederate

PingFederate is an enterprise federation server and identity bridge for user authentication and standards-based single sign-on (SSO) for employee, partner, and customer identities.

PingFederate enables outbound and inbound solutions for SSO, federated identity management, customer identity and access management, mobile identity security, API security, and social identity integration. Browser-based SSO extends employee, customer, and partner identities across domains without passwords using standard identity protocols, such as SAML, WS-Federation, WS-Trust, OAuth, OpenID Connect (OIDC), and System for Cross-domain Identity Management (SCIM).



About identity federation and SSO

Federated identity management, or identity federation, allows enterprises to exchange identity information securely across domains, providing browser-based single sign-on (SSO).

Identity federation also integrates access to applications across distinct business units within a single organization. As organizations grow through acquisitions, or when business units maintain separate user repositories and authentication mechanisms across applications, a federated solution to browser-based SSO is desirable.

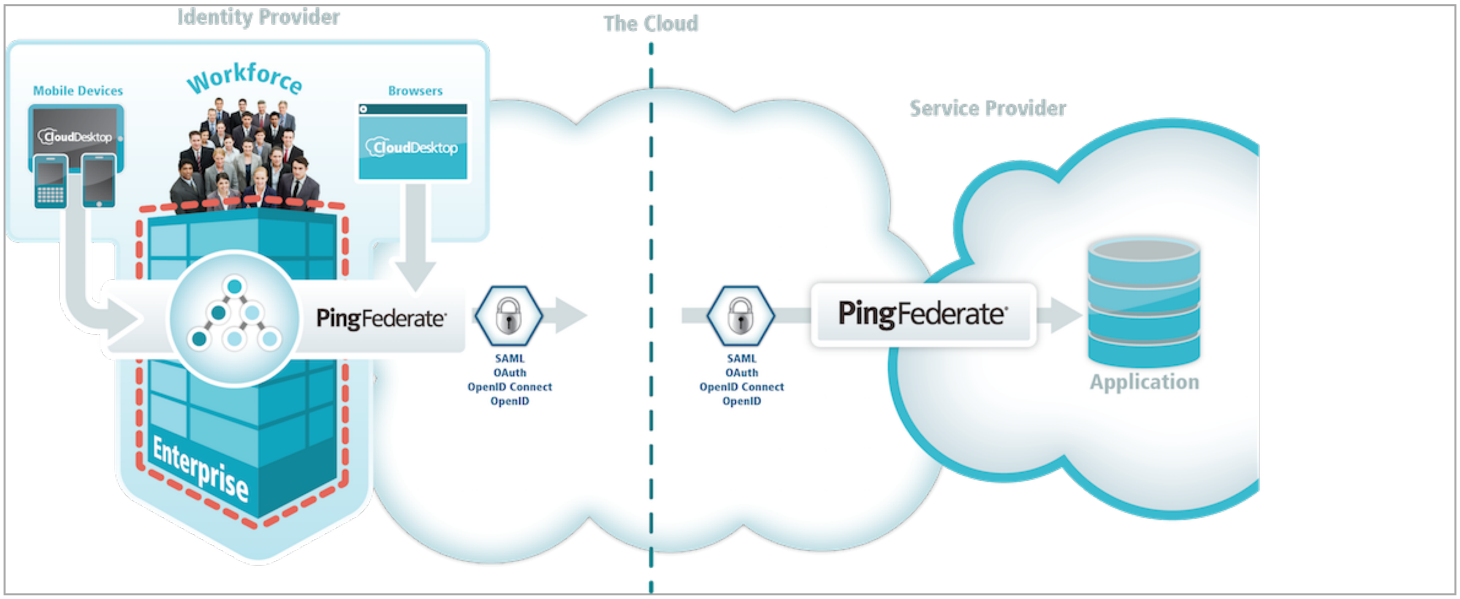
This cross-domain, identity-management solution provides numerous benefits, ranging from increased end user satisfaction and enhanced customer relations to reduced cost and greater security and accountability.

For complete information about identity federation and the standards that support it, see Supported standards.

Service providers and identity providers

Identity federation standards identify two operational roles in an SSO transaction: the identity provider (IdP) and the service provider (SP).

An IdP might be an enterprise that manages accounts for a large number of users who need secure access to the web-based applications or services of customers, suppliers, and business partners. An SP might be a SaaS provider or a business-process outsourcing (BPO) vendor wanting to simplify client access to its services.



Secure single sign-on

Identity federation allows both types of organizations to define a trust relationship whereby the SP provides access to users from the IdP. The IdP continues to manage its users, and the SP trusts the IdP to authenticate them.

A single instance of PingFederate provides complete support for both roles even when a single organization's business processes encompass both SP and IdP use cases.

Federation hub

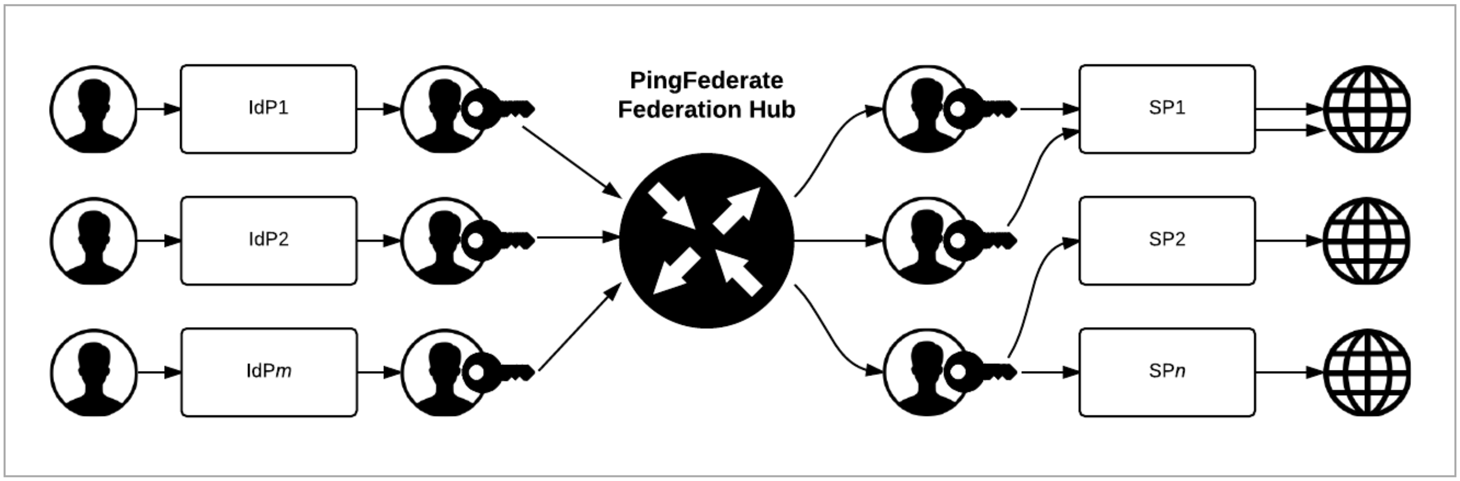
As a federation hub, PingFederate can bridge browser-based SSO between IdPs and SPs, reducing administrative overhead.

Identity federation refers to the negotiation and management of federation settings with partners. Supporting different federation protocols can hinder application development and SSO implementation.

Configuring PingFederate as a federation hub:

- Allows you to simplify application development and SSO implementation by extending federated access across partners supporting different federation standards
- Provides a centralized console to simplify SSO administration

Bridging IdPs and SPs through a federation hub allows you to multiplex a single connection for multiple partners.



Related links

- Federation hub use cases

Supported standards

PingFederate supports a variety of federation roles, protocols, and standards.

PingFederate provides flexible, integrated support for the SAML protocols, WS-Federation, OAuth, OpenID Connect, and WS-Trust. In addition, PingFederate supports System for Cross-domain Identity Management (SCIM) for inbound and outbound provisioning.

Federation roles

A variety of federation roles work together in an identity federation partnership.

The most recent sets of standards, SAML 2.0 and WS-Federation, define two roles in an identity federation partnership: an identity provider (IdP) and a service provider (SP).

> **Note**
>
> Earlier SAML 1.x specifications used the terms asserting party (for IdP) and relying party (for SP). For consistency and clarity, PingFederate adopts the later terms IdP and SP across all specifications.

A third role, defined in the SAML 2.0 specifications and available in PingFederate, is that of an IdP Discovery provider.

OAuth 2.0 and OpenID Connect 1.0 can configure PingFederate as an authorization server (AS), an OpenID provider (OP), and a relying party (RP).

Identity provider

An IdP, also called the SAML authority, is a system entity that authenticates a user, or SAML subject, and transmits referential identity information based on the authentication.

Note

The SAML subject may be a person, a web application, or a web server. Since the SAML subject is often a person, our documentation employs the term "user" throughout.

Service provider

An SP is the consumer of identity information provided by the IdP. Based on trust, technical agreements, and verification of adherence to protocols, SP applications and systems determine how to use information contained in an SSO token: a SAML assertion, a JSON Web Token (JWT), or an OAuth access token in conjunction with an ID token.

IdP Discovery provider

This role provides an IdP look-up service that can be incorporated into the implementation of either an IdP or an SP, or employed as a standalone server.

Authorization server

An OAuth authorization server issues access tokens and refresh tokens to OAuth clients after the resource owner fulfills the authentication requirement.

OpenID provider

An OpenID provider (OP) is an AS that is capable of authenticating the resource owner and providing claims (user attributes) to an RP about the authentication event and the user.

Terminology

Definitions for the list of SAML specifications that provide a system of building blocks and support components for achieving secure data exchange in an identity federation.

The list of SAML specifications includes:

- Assertions
- Bindings
- Profiles
- Metadata
- Authentication Context

Assertions

Assertions are XML documents sent from an identity provider (IdP) to a service provider (SP). Each assertion contains identifying information about a user who has initiated a single sign-on (SSO) request.

Bindings

A SAML binding describes the way transport protocols exchange messages. PingFederate supports the following bindings:

HTTP POST

Describes how to transport SAML messages in HTML form-control content, which uses a base-64 format.

HTTP Artifact

Describes how to use an artifact to represent a SAML message. An HTML form control or a query string in the URL transports the artifact.

HTTP Redirect (SAML 2.0)

Describes how to transport SAML messages using HTTP 302 status-code response messages.

SOAP (SAML 2.0)

Describes how to transfer SAML messages across the back channel.

Profiles

Profiles describe processes and message flows combining assertions, request/response message specifications, and bindings to achieve a specific desired functionality or use case. Profiles define the application of the specifications and play a large part in PingFederate.

Metadata

SAML 2.0 defines an XML schema to standardize metadata to facilitate the exchange of configuration information among federation partners, such as profile and binding support, connection endpoints, and certificate information.

Whether you publish or consume metadata, PingFederate supports the use of XML digital signatures to ensure the integrity of the data.

Authentication context

Before allowing access to a protected resource, an SP might want information about the assertion and the user's original authentication by the IdP. The SP uses this information for an access control decision or to provide an audit trail for regulatory or security-policy compliance.

The SAML 2.0 specification provides an XML schema whereby partners create authentication-context declarations. Partners might choose to reference a URI to implement a set of classes provided by the specification to help categorize and simplify context interpretation (see the OASIS document: Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0 ↗). However, it is up to partners to decide if they require additional authentication context and if these classes supply an adequate description. For SAML 1.x, if using the authentication context, called AuthenticationMethod, the context must appear as a URI (see).

An administrator can configure PingFederate, acting as an IdP, to include a specific authentication context in assertions for browser SSO or WS-Trust.

Several PingFederate integration kits provide methods for the developer to insert authentication context from external IdP applications into the assertion. The SP developer has the option to:

- Call methods for extracting authentication context from an assertion.
- Work with the application to create access control or other processing based on the context.

Browser-based SSO

Browser-based single sign-on (SSO) includes SAML 1.x profiles, SAML 2.0 profiles, WS-Federation, and OpenID Connect and provides standards-based SSO, single logout (SLO), attribute query and X.509 attribute sharing profile (XASP), and the WS-Federation Passive Requestor Profile for service provider (SP)-initiated SSO.

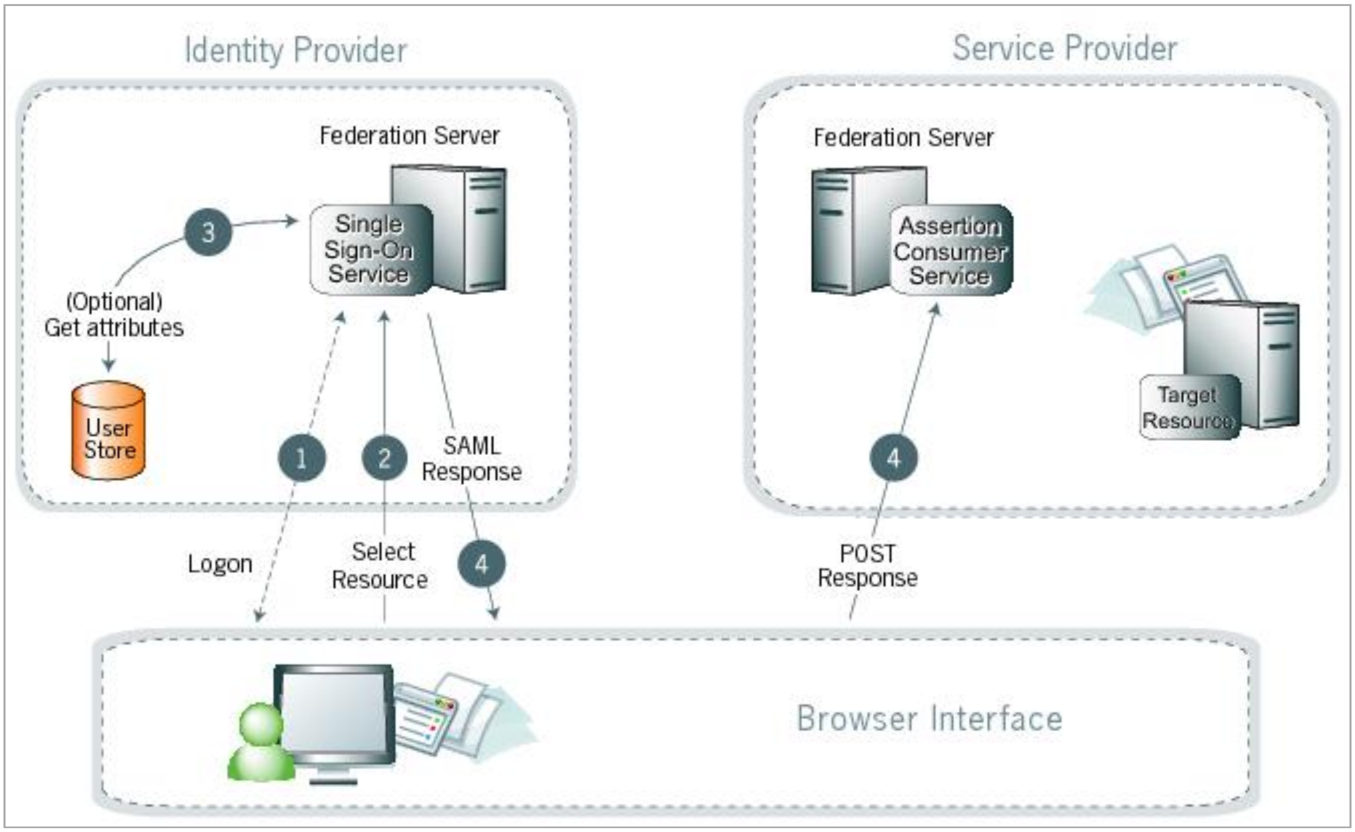
SAML 1.x profiles

SAML 1.0 and 1.1 profiles provide for browser-based single sign-on (SSO), initiated by an identity provider (IdP), using either the POST or artifact bindings.

The SAML 1.x specifications provide for a non-normative service provider (SP)-initiated scenario called "destination-first." This scenario lets web developers create applications that enable a user to initiate SSO from the SP site.

SSO—Browser-POST

In this scenario, a user logged on to the identity provider (IdP) attempts to access a resource on a remote service provider (SP) server. HTTP POST transports the SAML assertion to the SP.



SSO—Browser-POST profile

Processing steps

1. A user logs on to the IdP.

   If a user is not logged on for some reason, the IdP challenges them to do so at step 2.

2. The user clicks a link or otherwise requests access to a protected SP resource.

3. Optionally, the IdP retrieves attributes from the user data source.

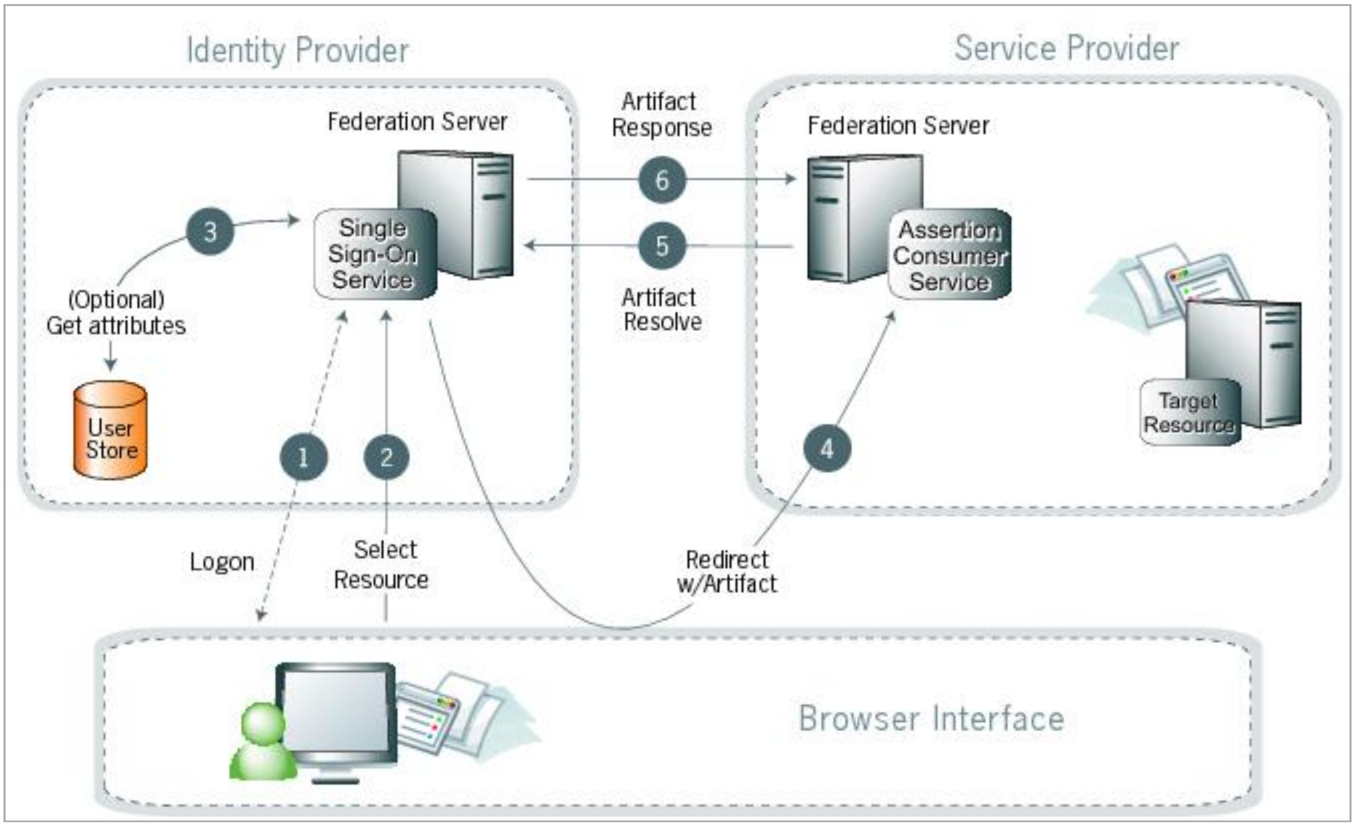
4. The IdP's SSO service returns an HTML form to the browser with a SAML response containing the authentication assertion and any additional attributes. The browser automatically posts the HTML form back to the SP.

   > **Note**
   >
   > SAML specifications require digitally-signed POST responses.

5. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

SSO—Browser-Artifact

In this scenario, the identity provider (IdP) sends a SAML artifact to the service provider (SP) through either HTTP POST or a redirect. The SP uses the artifact to obtain the associated SAML response from the IdP.



SSO—Browser-Artifact profile

Processing steps

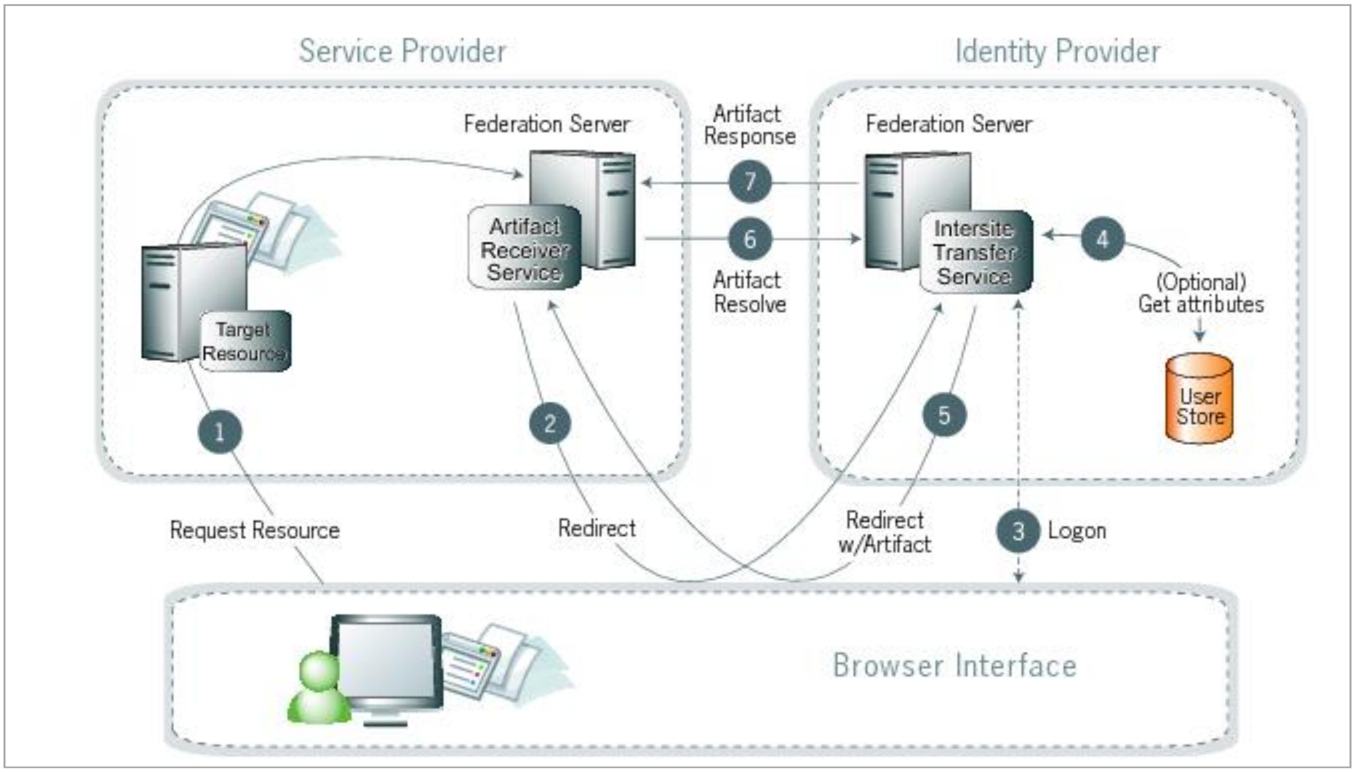
1. A user logs on to the IdP.

   If a user is not logged on for some reason, the IdP challenges them to do so at step 2.

2. The user clicks a link or otherwise requests access to a protected SP resource.
3. Optionally, the IdP retrieves attributes from the user datastore.
4. The IdP federation server generates an assertion, creates an artifact, and sends an HTTP redirect containing the artifact through the browser to the SP's Assertion Consumer Service (ACS).
5. The ACS extracts the Source ID from the SAML artifact and sends an artifact-resolve message to the identity federation server's Artifact Resolution Service (ARS).
6. The ARS sends a SAML artifact response message containing the previously-generated assertion.
7. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

SP-initiated (destination-first) SSO

In service provider (SP)-initiated, destination-first transactions, the user connects to an SP site and attempts to access a protected resource in the SP domain. The user might have an account at the SP site, but according to the federation agreement, the identity provider (IdP) manages authentication. The SP sends an authentication request to the IdP.



SP-initiated SSO

Processing steps

1. The user requests access to a protected SP resource. The request redirects to the federation server to handle authentication.
2. The federation server sends a SAML request for authentication to the IdP's single sign-on (SSO) service, also called the Intersite Transfer Service.
3. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.
4. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.
5. The IdP's Intersite Transfer Service returns an artifact representing the SAML response to the SP.
6. The SP's artifact handling service sends a SOAP request with the artifact to the IdP's artifact resolver endpoint.
7. The IdP resolves the artifact and returns the corresponding SAML response with the SSO assertion.
8. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

SAML 2.0 profiles

PingFederate supports the following major profiles defined under the SAML 2.0 standard:

- Single sign-on (SSO)
- Single logout (SLO)
- Attribute Query and XASP
- Standard IdP Discovery

Single sign-on

Pairing service provider (SP)- and identity provider (IdP)-initiated protocols with transport-binding specifications results in eight practical SSO scenarios.

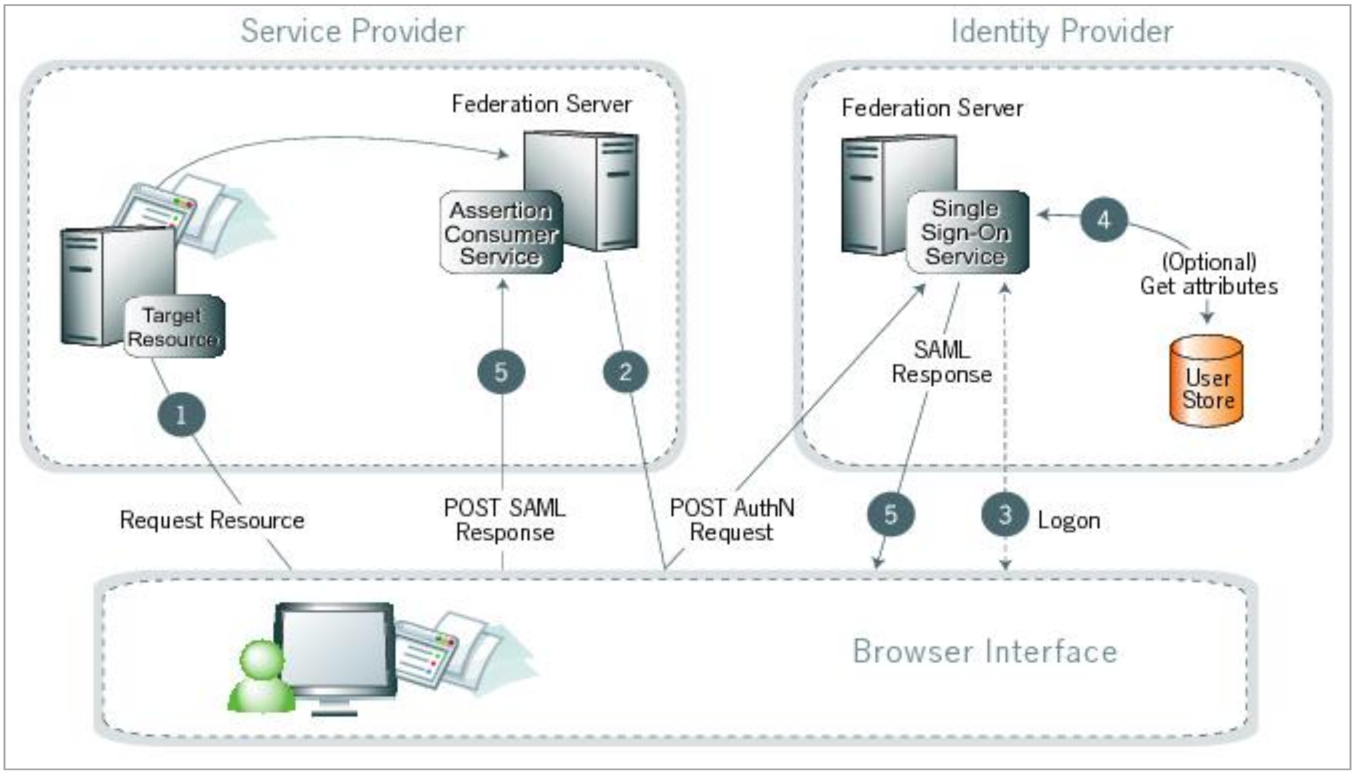
Enabling SP-initiated transactions through SAML 2.0 increases the number of possible SSO profile variations. The following profile variations each illustrate a specific scenario:

- SP-initiated SSO—POST-POST A user attempts to access a protected resource directly on an SP website without logging on. The user does not have an account on the SP site but does have a third-party IdP-federated account. The SP sends an authentication request to the IdP. The user's browser sends both the request and the returned SAML assertion through HTTP POST.
- SP-initiated SSO—Redirect-POST The SP sends an HTTP redirect message to the IdP containing an authentication request. The IdP returns a SAML response with an assertion to the SP via HTTP POST

- SP-initiated SSO—Artifact-POST The SP sends a SAML artifact to the IdP through an HTTP redirect. The IdP uses the artifact to obtain an authentication request from the SP's SAML artifact resolution service. The IdP returns a SAML response to the SP via HTTP POST.
- SP-initiated SSO—POST-Artifact the SP sends an authentication request to the IdP through HTTP POST. The returned SAML assertion is redirected through the user's browser. The response contains a SAML artifact.
- SP-initiated SSO—Redirect-Artifact The SP sends an HTTP redirect message to the IdP containing a request for authentication. The IdP returns an artifact through HTTP redirect. The SP uses the artifact to obtain the SAML response.
- SP-initiated SSO—Artifact-Artifact The SP sends a SAML artifact to the IdP through an HTTP redirect. The IdP uses the artifact to obtain an authentication request from the SP, and then the IdP sends another artifact to the SP, which the SP uses to obtain the SAML response.
- IdP-initiated SSO—POST A user is logged on to the IdP and attempts to access a resource on a remote SP server. The SAML assertion is transported to the SP through HTTP POST.
- IdP-initiated SSO—Artifact The IdP sends a SAML artifact to the SP through an HTTP redirect. The SP uses the artifact to obtain the associated SAML response from the IdP.

SP-initiated SSO—POST-POST

In this scenario, a user attempts to access a protected resource directly on a service provider (SP) website without logging on. The user does not have an account on the SP site but does have a third-party identity provider (IdP)-federated account. The SP sends an authentication request to the IdP. The user's browser sends both the request and the returned SAML assertion through HTTP POST.



SP-initiated SSO—POST-POST

Processing steps

1. The user requests access to a protected SP resource. The request redirects to the federation server to handle authentication.

2. The federation server sends an HTML form back to the browser with a SAML request for authentication from the IdP. The HTML form automatically posts to the IdP's SSO service.

3. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.

4. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.

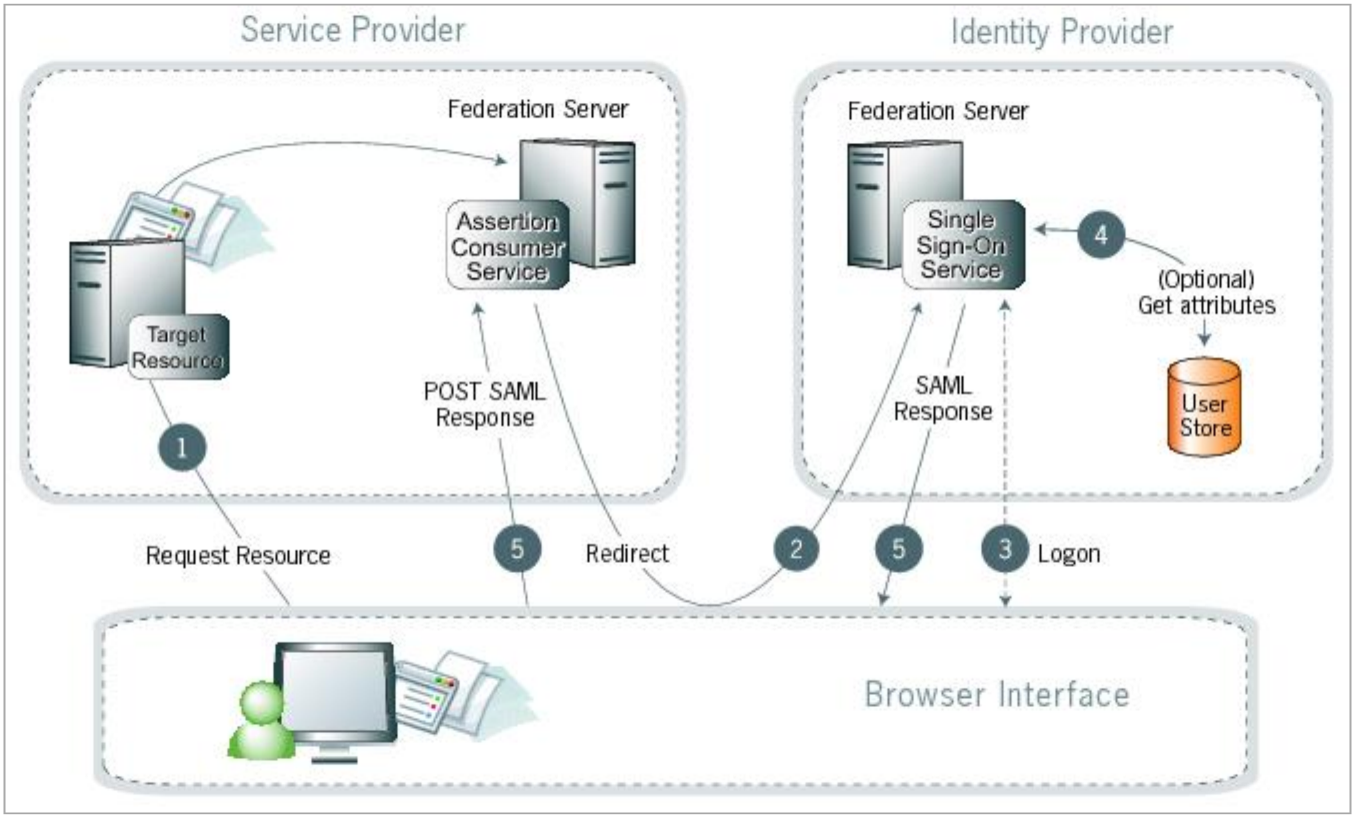
5. The IdP's SSO service returns an HTML form to the browser with a SAML response containing the authentication assertion and any additional attributes. The browser automatically posts the HTML form back to the SP.

> **Note**
>
> SAML specifications require digitally-signed POST responses.

6. (Not shown) If the signature and the assertion, or the JSON Web Token, are valid, the SP establishes a session for the user and redirects the browser to the target resource.

SP-initiated SSO—Redirect-POST

In this scenario, the service provider (SP) sends an HTTP redirect message to the identity provider (IdP) containing an authentication request. The IdP returns a SAML response with an assertion to the SP through HTTP POST.



SP-initiated SSO: redirect/POST

Processing steps

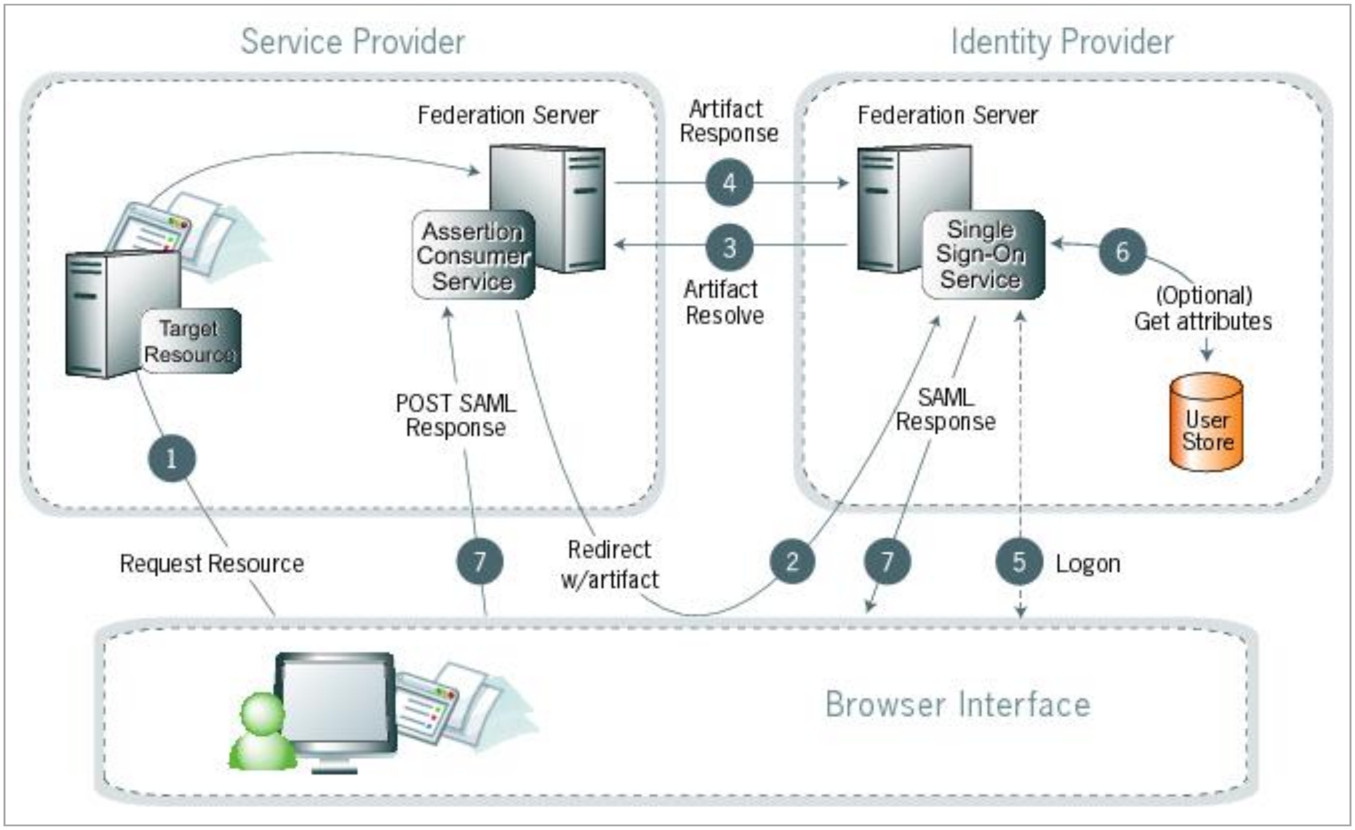
1. A user requests access to a protected SP resource. The user is not logged on to the site. The request redirects to the federation server to handle authentication.
2. The SP returns an HTTP redirect code, 302 or 303, containing a SAML request for authentication through the user's browser to the IdP's single sign-on (SSO) service.
3. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.
4. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.
5. The IdP's SSO service returns an HTML form to the browser with a SAML response containing the authentication assertion and any additional attributes. The browser automatically posts the HTML form back to the SP.

> **Note**
>
> SAML specifications require digitally-signed POST responses.

6. (Not shown) If the signature and the assertion, or the JSON Web Token, are valid, the SP establishes a session for the user and redirects the browser to the target resource.

SP-initiated SSO—Artifact-POST

In this scenario, the service provider (SP) sends a SAML artifact to the identity provider (IdP) through an HTTP redirect. The IdP uses the artifact to obtain an authentication request from the SP's SAML artifact resolution service (ARS), and the IdP returns a SAML response to the SP through HTTP POST.



SP-initiated SSO—Artifact-POST

Processing steps

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request redirects to the federation server to handle authentication.
2. The SP generates an authentication request and creates an artifact. The SP sends an HTTP redirect containing the artifact through the user's browser to the IdP's single sign-on (SSO) service.

> **Note**
>
> The artifact contains the source ID of the SP's artifact resolution service and a reference to the authentication.

3. The SSO service extracts a source ID from the SAML artifact and sends a SAML artifact-resolve message over SOAP containing the artifact to the SP's ARS.

> ⓘ **Note**
>
> The federation agreement made prior to this action maps the SP and IdP’s source IDs and remote ARS.

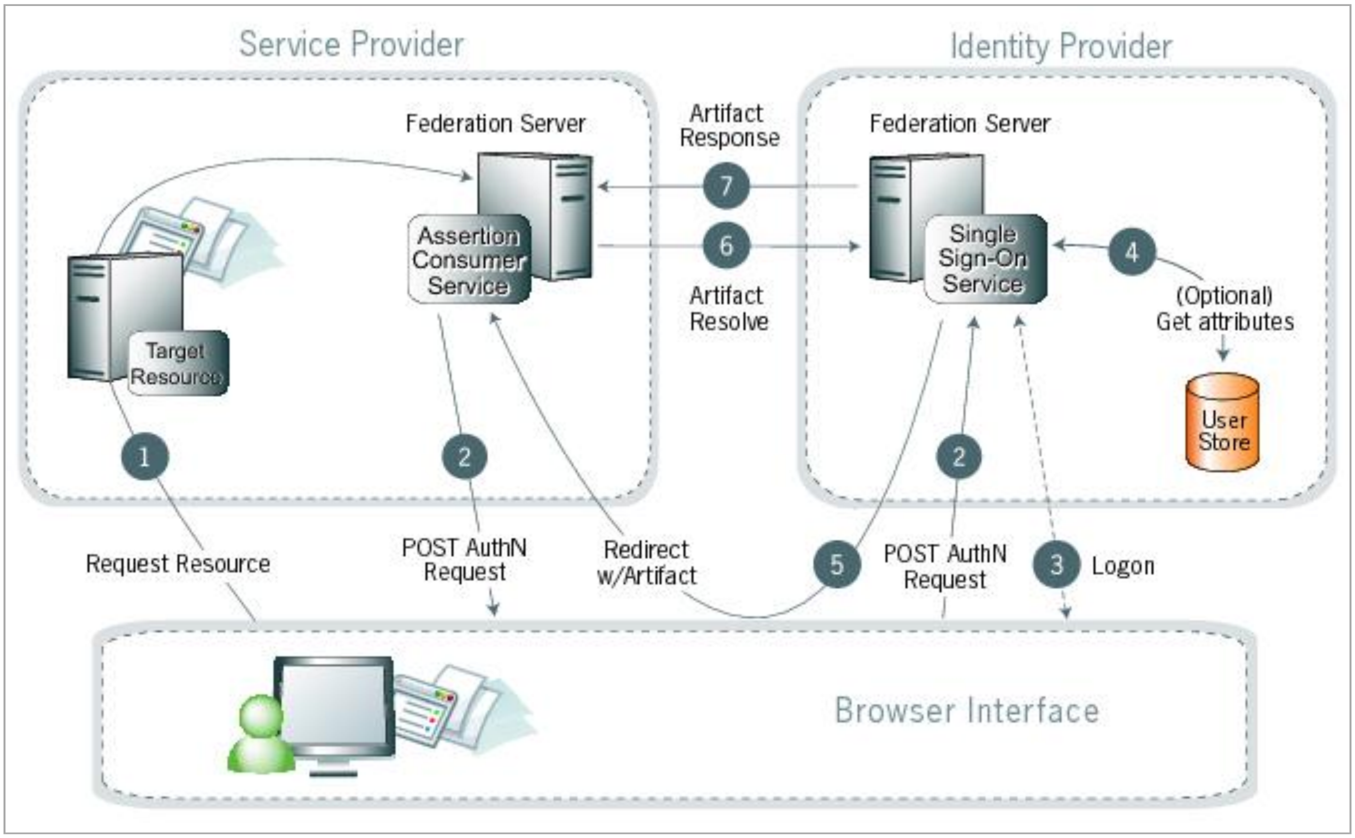
4. The SP’s ARS returns a SAML message containing the previously-generated authentication request.
5. If the user is not already logged on to the IdP site or needs to re-authenticate, The IdP asks for credentials, such as ID and password, and the user logs on.
6. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.
7. The IdP’s SSO service returns an HTML form to the browser with a SAML response containing the authentication assertion and any additional attributes. The browser automatically posts the HTML form back to the SP.

> ⓘ **Note**
>
> SAML specifications require digitally-signed POST responses.

8. (Not shown) If the signature and the assertion, or the JSON Web Token, are valid, the SP establishes a session for the user and redirects the browser to the target resource.

SP-initiated SSO—POST-Artifact

In this single sign-on (SSO) scenario, the service provider (SP) sends an authentication request to the identity provider (IdP) through HTTP POST. The returned SAML assertion redirects through the user’s browser, and the response contains a SAML artifact.



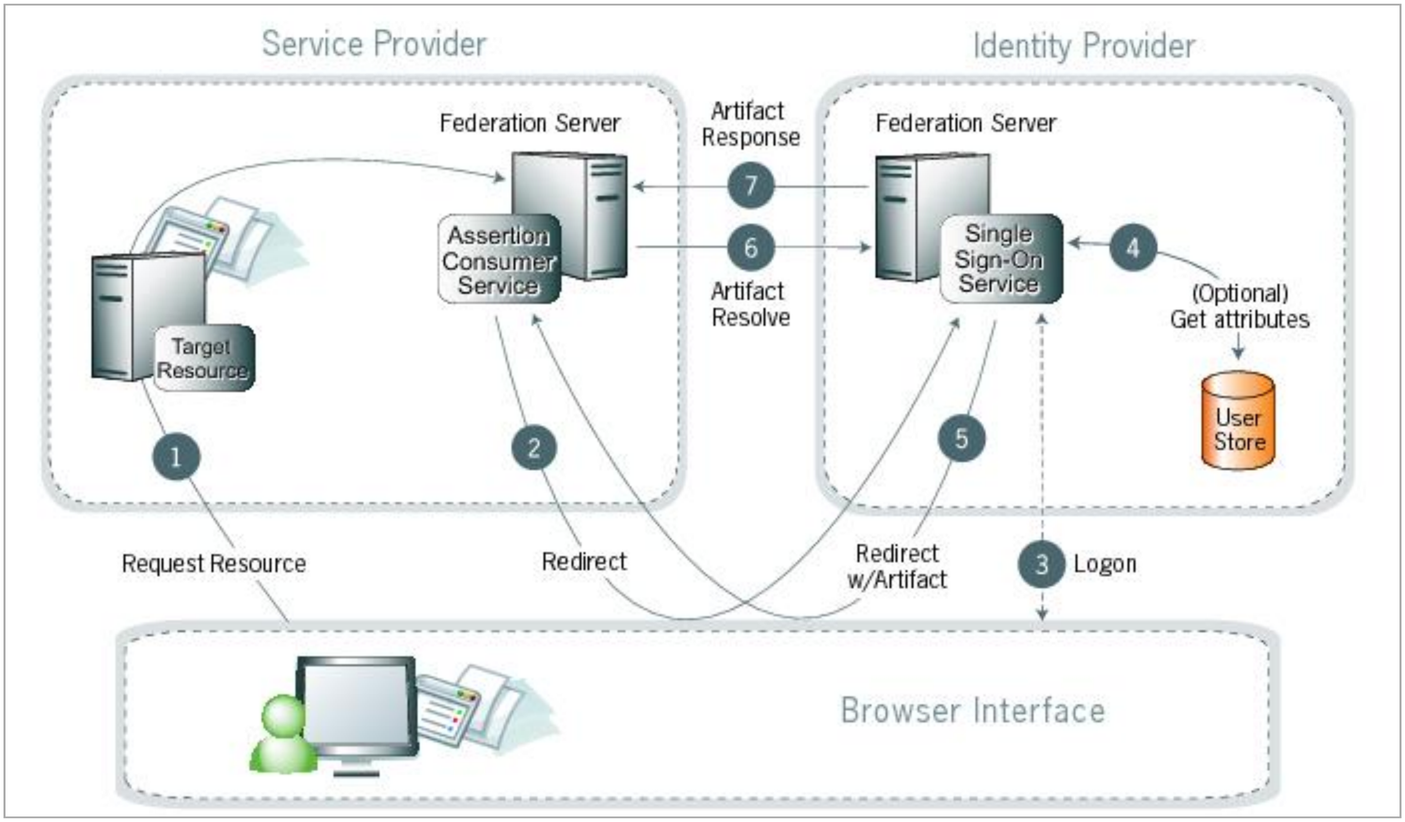
SP-initiated SSO—POST-Artifact

Processing steps

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request redirects to the federation server to handle authentication.
2. The federation server sends an HTML form back to the browser with a SAML request for authentication from the IdP. The HTML form automatically posts to the IdP’s SSO service.
3. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.
4. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.
5. The IdP federation server generates an assertion, creates an artifact, and sends an HTTP redirect containing the artifact through the browser to the SP’s Assertion Consumer Service (ACS).
6. The ACS extracts the source ID from the SAML artifact and sends an artifact-resolve message to the federation server’s Artifact Resolution Service (ARS).
7. The ARS sends a SAML artifact response message containing the previously-generated assertion.
8. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

SP-initiated SSO—Redirect-Artifact

In this scenario, the service provider (SP) sends an HTTP redirect message to the identity provider (IdP) containing a request for authentication. The IdP returns an artifact through HTTP redirect, and the SP uses the artifact to obtain the SAML response.



SP-initiated SSO—Redirect-Artifact

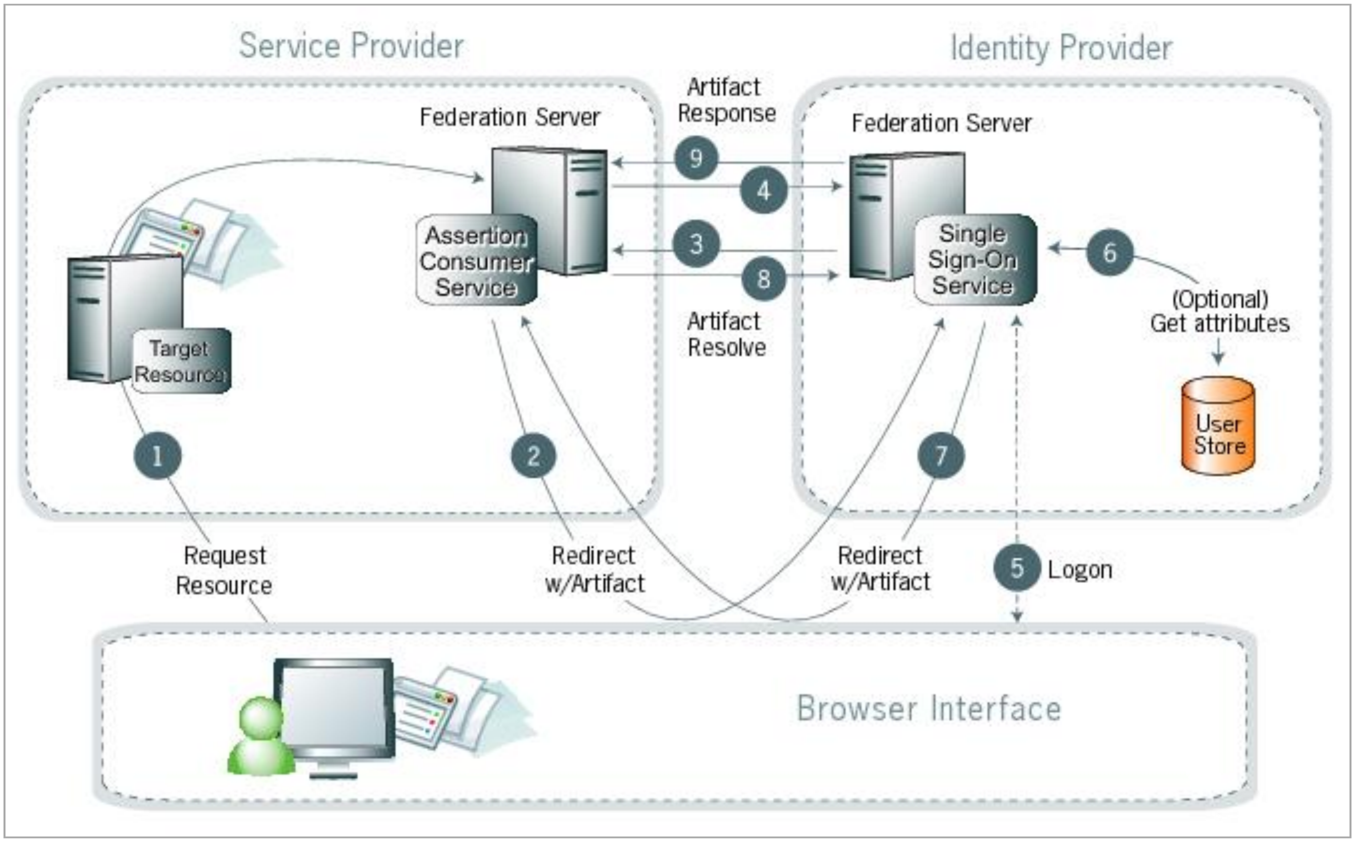
Processing steps

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request redirects to the federation server to handle authentication.
2. The SP returns an HTTP redirect, either code 302 or 303, containing a SAML request for authentication through the user's browser to the IdP's single sign-on (SSO) service.
3. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.
4. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.
5. The IdP federation server generates an assertion, creates an artifact, and sends an HTTP redirect containing the artifact through the browser to the SP's Assertion Consumer Service (ACS).
6. The ACS extracts the Source ID from the SAML artifact and sends an artifact-resolve message to the identity federation server's Artifact Resolution Service (ARS).
7. The ARS sends a SAML artifact response message containing the previously-generated assertion.

8. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

SP-initiated SSO—Artifact-Artifact

In this scenario, the service provider (SP) sends a SAML artifact to the identity provider (IdP) through an HTTP redirect. The IdP uses the artifact to obtain an authentication request from the SP, then the IdP sends another artifact to the SP, which the SP uses to obtain the SAML response.



SP-initiated SSO—Artifact-Artifact

Processing steps

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request redirects to the federation server to handle authentication.
2. The Assertion Consumer Service (ACS) generates an authentication request and creates an artifact. It sends an HTTP redirect containing the artifact through the user's browser to the IdP's single sign-on (SSO) service.

> **Note**
>
> The artifact contains the source ID of the SP's Artifact Resolution Service (ARS) and a reference to the authentication request.

3. The SSO service extracts the source ID from the SAML artifact and sends a SAML artifact resolve message containing the artifact to the SP's artifact resolution service.

   > **Note**
   >
   > The federation agreement maps the SP and IdP's source IDs and remote artifact resolution services prior to this action.

4. The SP's artifact resolution service sends back a SAML artifact response message containing the previously-generated authentication request.

5. If the user is not already logged on to the IdP site or needs to re-authenticate, the IdP asks for credentials, such as ID and password, and the user logs on.

6. The user data store can provide additional information about the user for inclusion in the SAML response. The federation agreement between the IdP and the SP predetermines these attributes. See User attributes.

7. The IdP federation server generates an assertion, creates an artifact, and sends an HTTP redirect containing the artifact through the browser to the SP's Assertion Consumer Service (ACS).

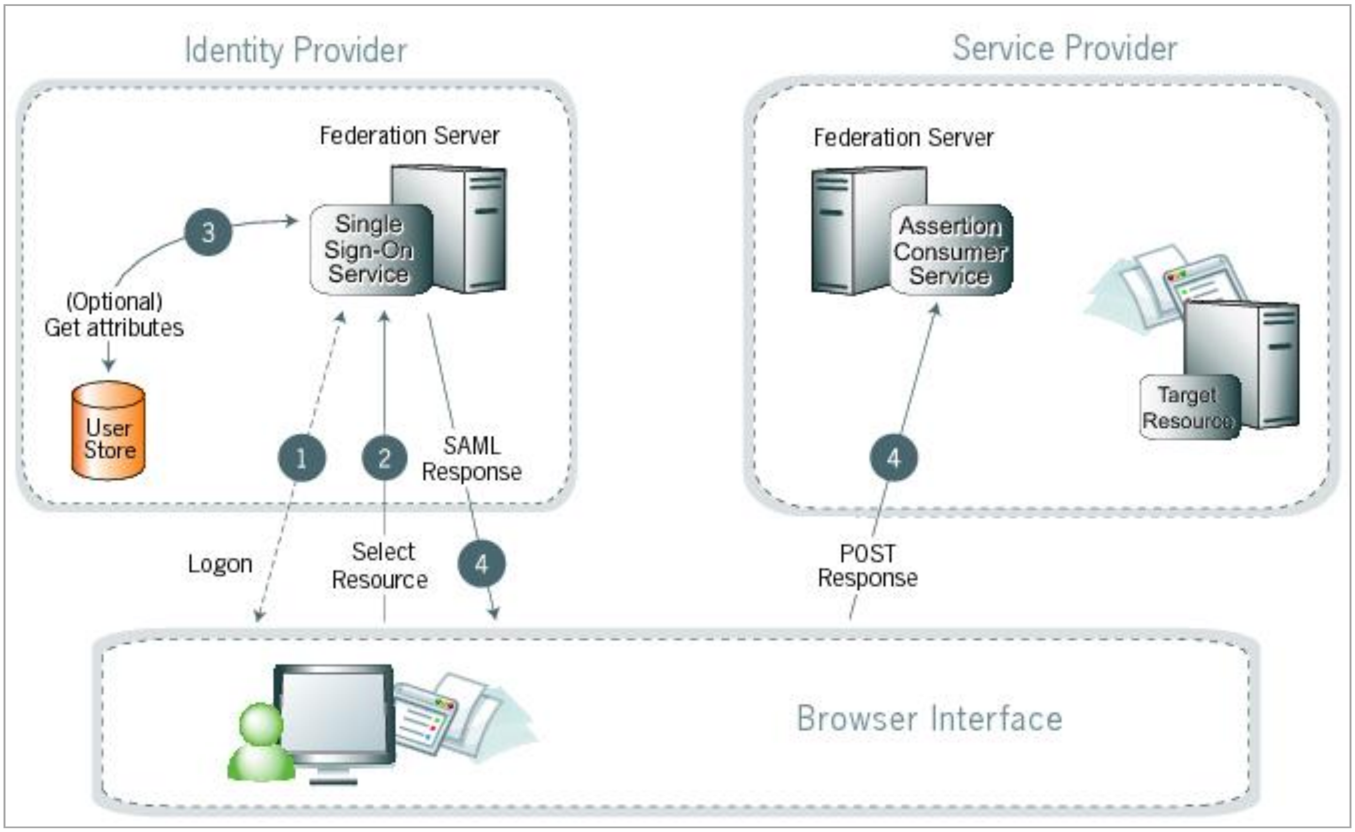
8. The ACS extracts the Source ID from the SAML artifact and sends an artifact-resolve message to the identity federation server's Artifact Resolution Service (ARS).

9. The ARS sends a SAML artifact response message containing the previously-generated assertion.

10. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

IdP-initiated SSO—POST

In this scenario, a user is logged on to the identity provider (IdP) and attempts to access a resource on a remote service provider (SP) server. HTTP POST transports the SAML assertion to the SP.



IdP-initiated SSO—POST

Processing steps

1. A user logs on to the IdP.

   If a user is not yet logged on for some reason, they are challenged to do so at step 2.

2. The user requests access to a protected SP resource.

3. After the user requests access, the IdP might also retrieve attributes from the user datastore..

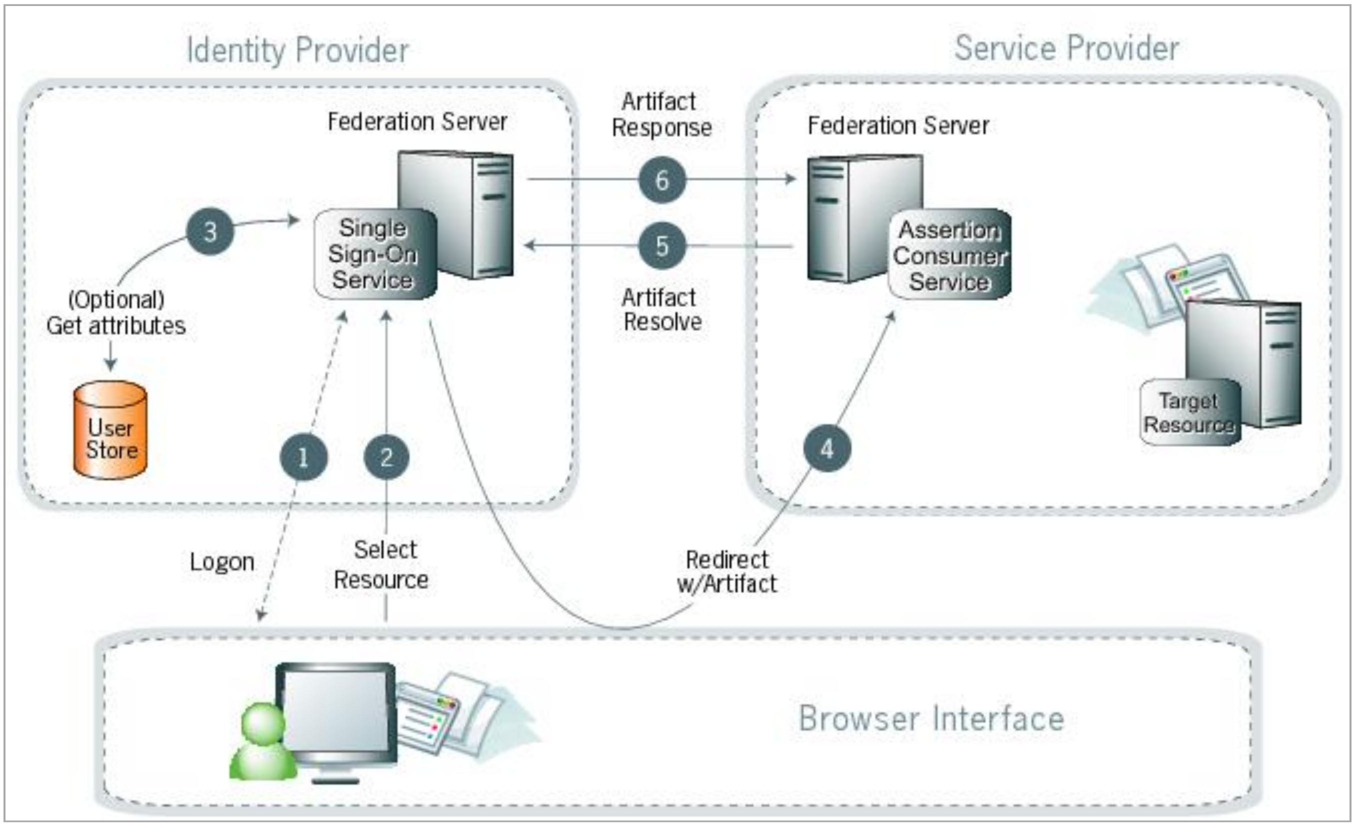
4. The IdP’s SSO service returns an HTML form to the browser with a SAML response containing the authentication assertion and any additional attributes. The browser automatically posts the HTML form back to the SP.

   > **Note**
   >
   > SAML specifications require digitally-signed POST responses.

5. (Not shown) If the signature and the assertion, or the JSON Web Token, are valid, the SP establishes a session for the user and redirects the browser to the target resource.

IdP-initiated SSO—Artifact

In this single sign-on (SSO) scenario, the identity provider (IdP) sends a SAML artifact to the service provider (SP) through an HTTP redirect. The SP uses the artifact to obtain the associated SAML response from the IdP.



IdP-initiated SSO—Artifact

Processing steps

1. A user logs on to the IdP.

   If a user has not yet logged on for some reason, they are challenged to do so at step 2.

2. The user clicks a link or otherwise requests access to a protected SP resource.

3. After the user requests access, the IdP might also retrieve attributes from the user datastore.

4. The IdP federation server generates an assertion, creates an artifact, and sends an HTTP redirect containing the artifact through the browser to the SP's Assertion Consumer Service (ACS).

5. The ACS extracts the Source ID from the SAML artifact and sends an artifact-resolve message to the identity federation server's Artifact Resolution Service (ARS).

6. The ARS sends a SAML artifact response message containing the previously-generated assertion.

7. (Not shown) If the IdP returns a valid SAML assertion to the SP, a session is established on the SP and the browser is redirected to the target resource.

Single logout

The single logout (SLO) profile enables users to log out of all participating sites in a federated session from any site.

The associated identity provider (IdP) federation deployment manages all logout requests and responses for participating sites. If a participating site returns an error, other participating sites might not receive their logout requests. In this scenario, PingFederate returns an error message to the end users.

The logout messages can be transported using any combination of bindings described for SSO (POST, artifact, or redirect). See the diagrams under SAML 2.0 profiles for illustrations of these message flows.

About session cleanup

When a service provider (SP) receives an SLO request from an IdP, the session creation adapters must handle any session clean-up involving the local application.

Attribute Query and XASP

The SAML 2.0 Attribute Query profile allows a service provider (SP) to request user attributes from an identity provider (IdP) in a secure transaction separate from single sign-on (SSO).The X.509 Attribute Sharing Profile (XASP) defines a specialized extension of the general Attribute Query profile.

The IdP, acting as an attribute authority, accepts attribute queries, performs a datastore lookup into a user repository such as an LDAP directory, provides values to the requested attributes, and generates an attribute response back to the originating SP requester. The SP then returns the attributes to the requesting application.

> **Tip**
>
> When privacy is required for sensitive attributes, you can configure PingFederate to obfuscate, or mask, their values in the server and transaction logs.

Web SSO is distinct from the Attribute Query use case. You can configure PingFederate servers to implement either of these profiles without regard to the other.

The XASP specification enables organizations with an investment in Public Key Infrastructure (PKI) to issue and receive Attribute Queries based on user-certificate authentication.

Under XASP a user authenticates directly with an SP application by providing their X.509 certificate. After the user is authenticated, the SP application requests additional user attributes by contacting the SP PingFederate server. A portion of the user's X.509 certificate is included in the request and can be used to determine the correct IdP to use as the source of the requested attributes. Finally, the SP generates an Attribute Query and transmits it to the IdP over the SOAP back channel.

Because the user arrives at the SP server already authenticated, no PingFederate adapter is used in this case.

Related links

- Configuring the Attribute Query profile in an SP connection
- Manage the Attribute Query profile in an IdP connection

Standard IdP Discovery

SAML 2.0 identity provider (IdP) Discovery provides a cookie-based look-up, when the IdP is not otherwise specified, to dynamically identify a user's IdP during a service provider (SP)-initiated single sign-on (SSO) event.

This mechanism is helpful in cases where an SP is a hub for several IdPs in an identity federation.

> **Tip**
>
> In addition to supporting standard IdP Discovery, PingFederate provides a cross-protocol, proprietary mechanism that allows an SP server to write a persistent browser cookie. The cookie contains a reference to the previous IdP-authentication partner for SSO. For more information, see Configuring IdP discovery using a persistent cookie.

In the standard scenario, when a user requests access to a protected resource on the SP, common-domain browser cookies are used to determine where a user has previously authenticated. Using this information, a PingFederate server can determine which IdP connection to use for sending an authentication request.

PingFederate can serve in up to three different IdP Discovery provider roles: common domain server, common domain cookie writer, and common domain cookie reader. Each of these roles is necessary to support IdP Discovery. The roles can be distributed across multiple servers at different sites.

Common domain server

In this role, the PingFederate server hosts a domain that its federation partners share in common. The common domain server allows partners to manipulate browser cookies that exist within that common domain. PingFederate can serve in this role exclusively or as part of either an IdP or an SP federation role, or both.

Common domain cookie writer

When PingFederate is acting in an IdP role and authenticates a user, it can write an entry in the common domain cookie, including its federation entity ID. An SP can look up this information on the common domain (not the same location as the common domain server described above).

Common domain cookie reader

When PingFederate is acting as an SP and needs to determine past IdP authentications, it reads the common domain cookie. Based on the information contained in the cookie, PingFederate can then initiate an SSO authentication request using the correct IdP connection.

WS-Federation

PingFederate supports the WS-Federation Passive Requestor Profile for service provider (SP)-initiated single sign-on (SSO), enabling interoperability with Microsoft's Active Directory Federation Service (ADFS).

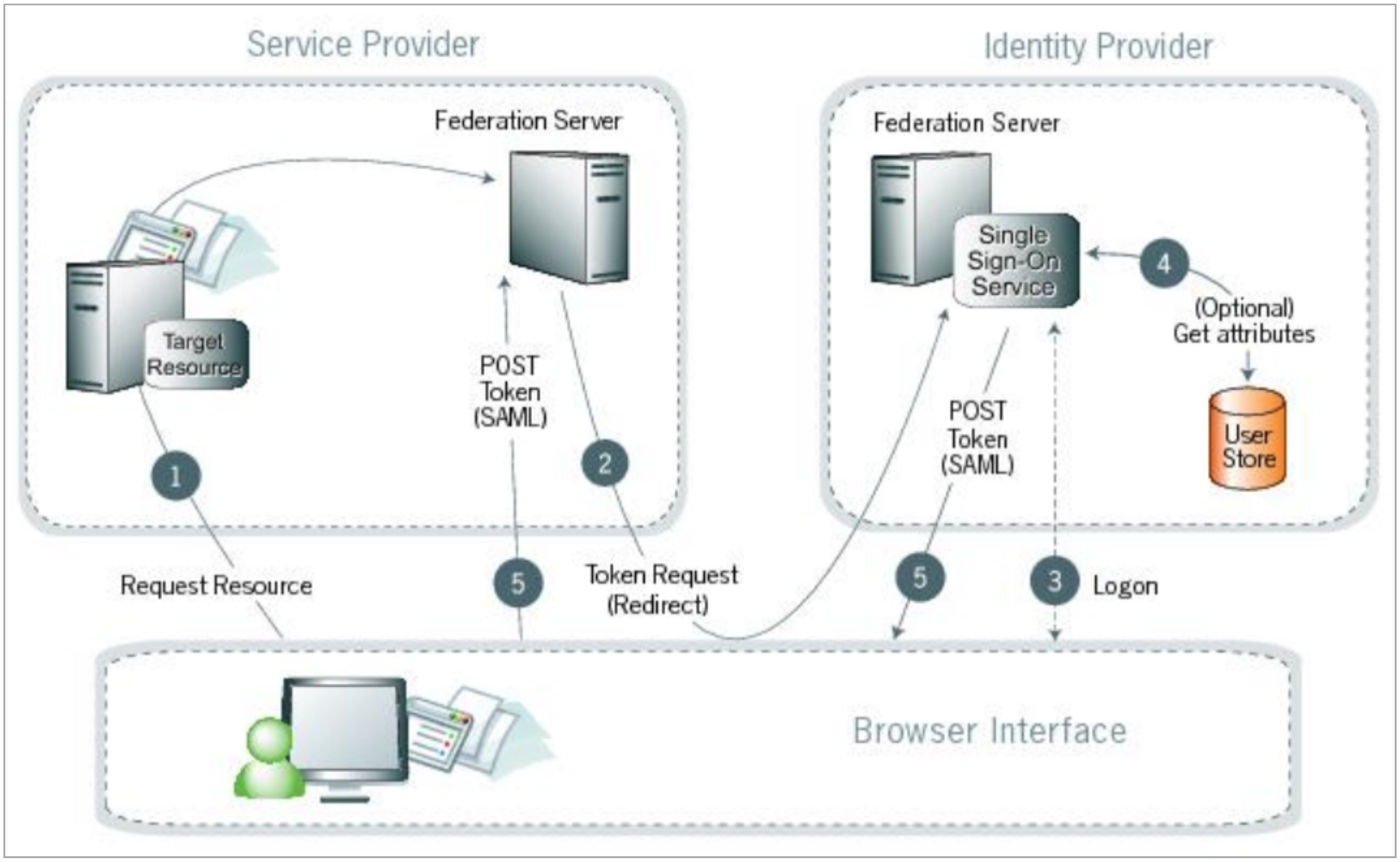
This profile allows for straightforward redirects and HTTP GET and POST methods to transport SAML assertions or JSON web tokens (JWTs) as security tokens for SSO and logout request and response messages for single logout (SLO).

> **Note**
>
> Unlike SAML, WS-Federation consolidates the endpoints for SLO and SSO. When you set up a WS-Federation connection in PingFederate, both types of transactions are available to an SP web application that supports them both.

For more information about WS-Federation and the Passive Requestor Profile, see Web Services Federation Languages (WS-Federation), Version 1.2.

Passive Requestor profile

This profile permits a user's browser, the passive requestor, to request a security token from an identity provider (IdP) when the user requests access to a protected web service or other resource at an SP.



WS-Federation SSO

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request is redirected to the federation server to handle authentication.
2. The SP generates a security token request and redirects the browser to the identity provider's WS-Federation implementation.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials, such as ID and password, and the user logs on.
4. Additional information about the user can be retrieved from the user datastore for inclusion in the SAML response. These attributes are predetermined as part of the federation agreement between the IdP and the SP. For more information, see User attributes.
5. The federation server creates a response containing a signed SAML assertion, or a JSON Web Token, and returns it to the SP through POST.
6. (Not shown) If the signature and the assertion, or the JWT, are valid, the SP establishes a session for the user and redirects the browser to the target resource.

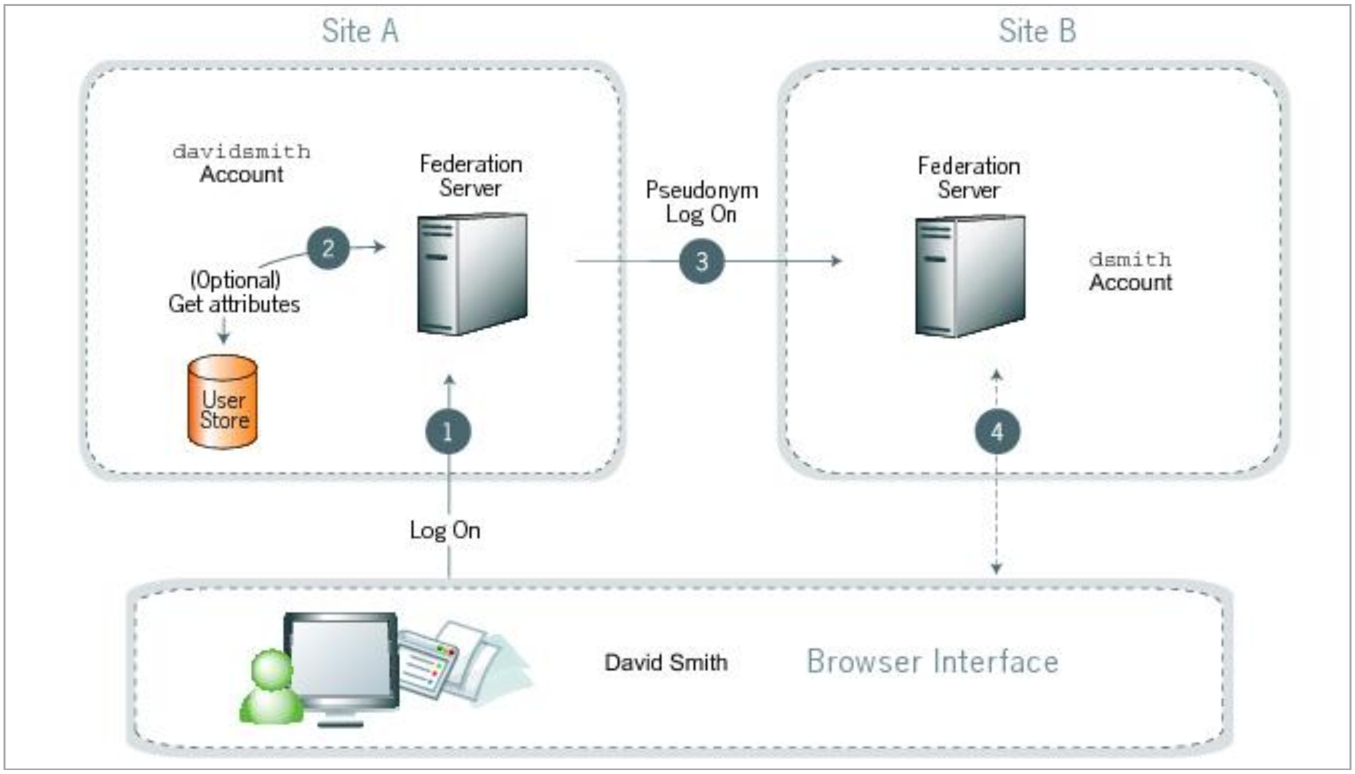
Single logout using WS-Federation is handled in much the same way as with SAML. For more information, see Single logout. However, HTTP GET/POST is always used as the transport mechanism.

About account linking

Account linking provides a means for a user to log on to disparate sites with just one authentication when the user has established accounts and credentials at each site.

All protocols support this method of interconnecting accounts across domains.

Account linking involves a persistent name identifier associated with accounts at each participating site. The assertion conveys the name identifier, which can be an opaque pseudonym. Once established locally, the service provider (SP) can use the account link to look up the user and provide access without re-authentication.



Account linking

Processing steps

1. David Smith logs on to Site A as *davidsmith*. He then decides to access his account on Site B through Site A.

2. Optionally, the federation server looks up additional attributes from the datastore.

3. The Site A federation server sends a persistent name identifier to Site B, along with any other attributes.

> **Note**
>
> When using a pseudonym and sending other attributes, be careful not to send attributes that could identify the subject.

4. The federation server on Site B uses the information to associate the pseudonym with the existing account of *dsmith* and optionally might ask David to provide consent to the linking.

   Once the link has been established, Site B stores the information so that David only has to log on to Site A to access Site B.

Web services standards

The PingFederate WS-Trust security token service (STS) interoperates with many different web-service environments that support varying standards.

PingFederate supports multiple versions of SOAP and WS-Trust specifications and operates with any combination of these standards simultaneously.

PingFederate supports namespace aliasing to eliminate common trailing-slash inconsistencies for WS-Trust 1.3. The server does not support namespace aliasing for WS-Trust 2005.

The following table lists supported SOAP/WS-Trust versions and corresponding namespaces.

SOAP/WS-Trust versions

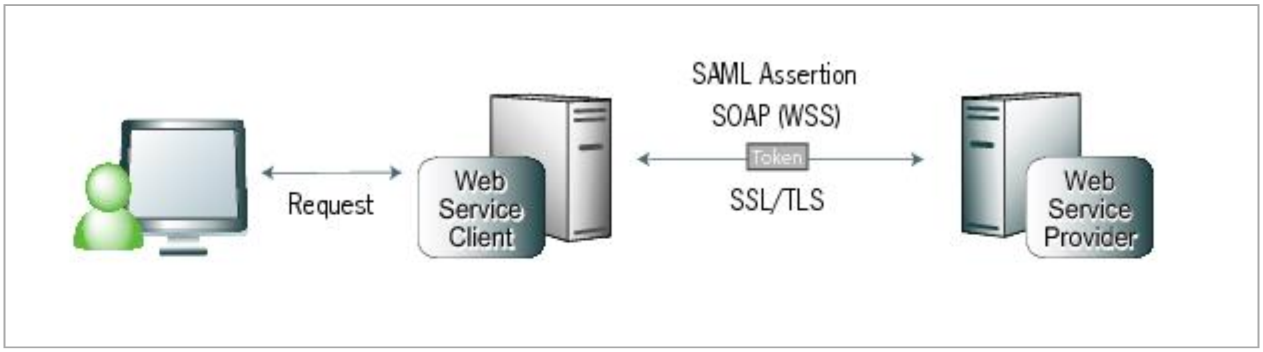
Spec	Version	Namespace
SOAP	1.1	http://schemas.xmlsoap.org/soap/envelope/
	1.2	http://www.w3.org/2003/05/soap-envelope
WS-Trust	2005	http://schemas.xmlsoap.org/ws/2005/02/trust/
	1.3	http://docs.oasis-open.org/ws-sx/ws-trust/200512/

Web Services Security

Web Services Security (WSS or WSSE) is a set of specifications defined by the OASIS Web Services Security (WSS) Technical Committee..

WSS defines XML extensions used to secure web service invocations, providing a standard way for partners to add message integrity and confidentiality to web service interactions. The WSS-defined token profiles describe standard ways of binding security tokens to these messages, enabling a variety of additional capabilities. Defined profiles include SAML assertions, Username, Kerberos, X.509, and other existing security tokens. SSL/TLS is often used in conjunction with deployments of WSS. For more information see https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

> **Note**
>
> The implementation of WSS in the deployment of web services identity federations is outside the scope of PingFederate, which provides a standalone, standard means of handling the tokens needed for such federations. See WS-Trust.



WSS token transfer

Processing steps

1. A user requests content from an application.
2. The web service client sends a web service request to the WSP, including the SAML assertion in a WSS header.
3. The WSP responds to the request and sends an SSL/TLS token back to the application.
4. The web service client returns an HTML page to the user.

WS-Trust

WS-Trust comprises a protocol that systems and applications use when requesting a service to issue, validate, and exchange security tokens.

Organizations can leverage WS-Trust to centralize their security-token processing. The WS-Trust specification defines the role of a Security Token Service (STS) as the entity responsible for responding to requests using the protocol. In this role, the STS creates new security tokens, validates existing security tokens, exchanges security tokens of one type for those of another, or any combination of these.

The WS-SX (Secure Exchange) technical committee manages the WS-Trust specification through its contribution to the OASIS standards organization. For more information, see www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx.

Request types

The WS-Trust protocol defines two request types in securing web services: Issue and Validate, often associated with the web service client (WSC) and web service provider (WSP), respectively.

- The WSC requests that a security token service (STS) issue a SAML token to convey information between the WSC and the WSP.
- The WSP sends the STS a request to validate the incoming token. Optionally, the WSP can request that the STS issue a local token for the service provider (SP) domain.

When issuing and validating security tokens, PingFederate enforces security policies, defined by administrators, generating the token types that are required for a web service request to pass between two security domains (whether these domains are within the same organization or in separate organizations).

Token exchange (example)

```
image:twb1564003103973.jpg[alt="Diagram illustrating a token exchange, using to obtain a SAML assertion. The
SAML assertion is used in the WSS-secured web service call."]
```

Processing steps

1. A user requests content from an application.
2. The application acts as a WSC to respond to the user's request. The application calls PingFederate, passing the existing user security token to exchange it for the appropriate SAML assertion.
3. PingFederate verifies the existing security token, creates a new SAML assertion representing the user, and returns it to the requesting application.
4. The application sends a web service request to the WSP, including the SAML assertion in a WS-Security header.
5. The WSP retrieves the SAML assertion from the WS-Security header in the incoming request and sends a message to its own deployment of PingFederate to determine if the assertion is valid.
6. PingFederate validates the SAML assertion, creates a new security token for the local domain, and returns the new token to the WSP.
7. The WSP responds to the request according to its policy for the user.
8. The web application returns an HTML page to the user.

> **Note**
>
> This example shows PingFederate deployed in both the WSC and WSP sides of the interaction. However, other deployment options are also supported.

OAuth 2.0 and PingFederate AS

OAuth 2.0 defines a protocol for securing application access to protected resources by issuing access tokens to clients of REST APIs and non-REST APIs.

Instead of the client directly authenticating to the API using credentials, or the credentials of a user, OAuth enables the client to authenticate by presenting a previously-obtained token. The token represents or contains a set of attributes, policies, or both appropriate to the client and the user. Using these tokens is more secure than using passwords directly on the API call. The attributes are used by the API to authenticate the call and authorize access.

Participants

Client

Wants access to a resource protected by a resource server and interacts with an authorization server to obtain access tokens.

Resource server (RS)

Hosts and protects resources and makes them available to authenticated and authorized clients.

Authorization server (AS)

Issues access tokens and refresh tokens to clients on behalf of the resource servers.

Resource owner (RO)

Denies, grants, or revokes authorization to a client requesting access to resources protected by the resource servers.

Tokens

Access token

Allows clients to authenticate to a resource server and claim authorizations for accessing particular resources. Access tokens have specific authorization scope and duration.

Refresh token

Allows clients to obtain a fresh access token without re-obtaining authorization from the resource owner. A refresh token is a long-lived token that a client can trade in to an authorization server to obtain a new short-lived access token with the same attached authorizations as the existing access token.

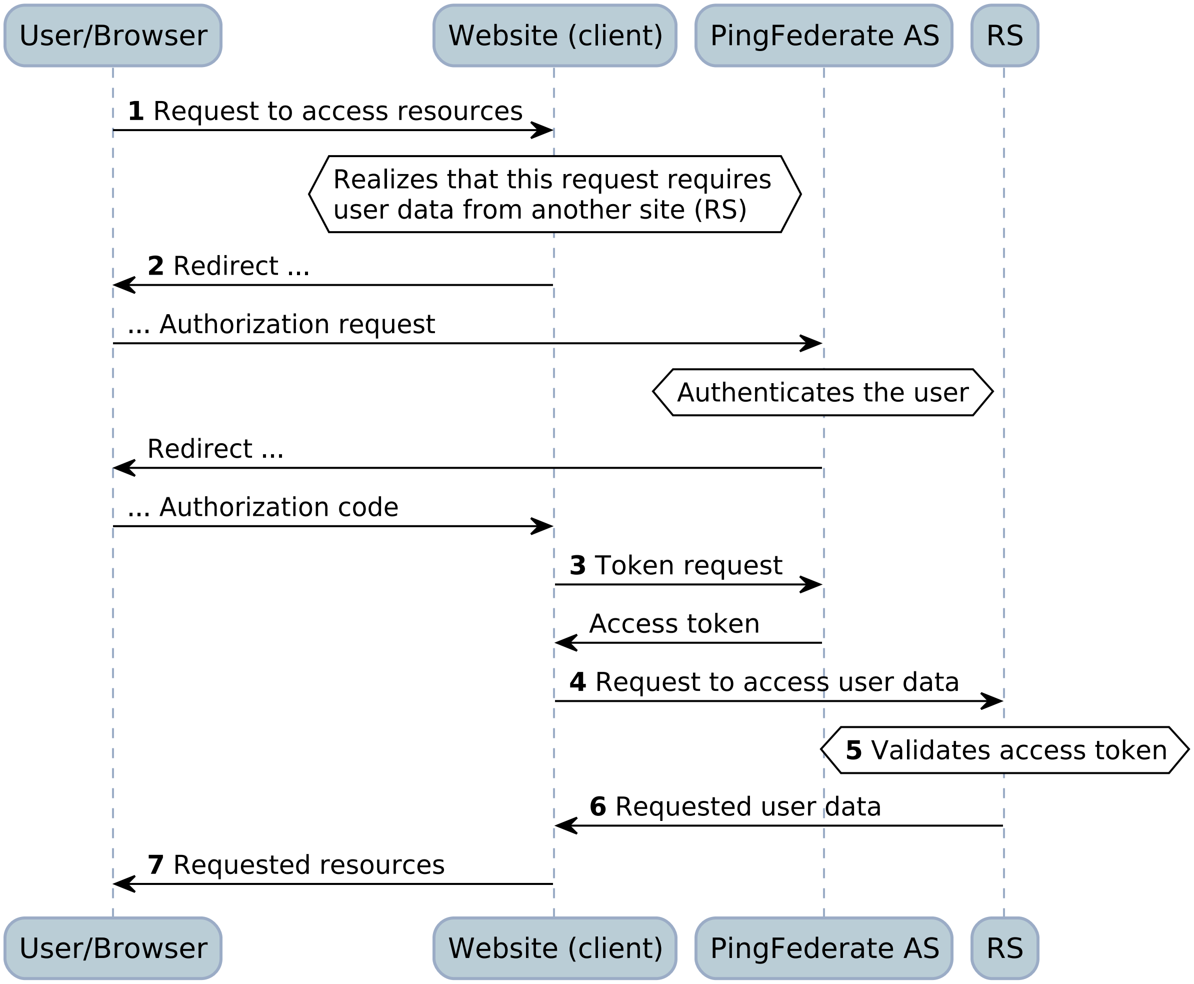
PingFederate OAuth AS

Based on the Internet Engineering Task Force (IETF) OAuth 2.0 Authorization Framework, the OAuth AS in PingFederate supports several interaction models for different types of clients such as servers, desktop applications, or mobile applications. Administrators can also enable Cross-origin Resource Sharing (CORS) support for OAuth endpoints.

Web redirect flow

The web redirect flow process takes place between the user, website, PingFederate authorization server (AS), and resource server (RS).

In this scenario, a user attempts to access a protected resource through a third-party web server client. The client sends an authorization request to the resource server, and receives an authorization code back through a HTTP redirect. The client trades the authorization code for an access token, and uses the token in an API call to obtain data.



Web redirect flow

Processing steps

1. User navigates to an OAuth client website and requests access to protected resources from another website. Flow chart depicting the process of web redirect flow between the User/Browser, Website (client), PingFederate AS and RS.

   > **Note**
   >
   > To reduce the risk of code interception attach, the OAuth client can optionally include the parameter `code_challenge` with or without `code_challenge_method` . For more information, see step 3 and Flow chart depicting the process of web redirect flow between the User/Browser,Proof Key for Code Exchange by (PKCE) OAuth Public Clients.

2. The browser is redirected to the PingFederate OAuth AS with a request for authorization.

   If the user is not logged on, the OAuth AS challenges the user to authenticate. The OAuth AS authenticates the user and prompts for authorization. After the user authorizes, the OAuth AS redirects the browser to the requesting site with an authorization code. If the user does not authenticate, the OAuth AS returns an error rather than the authorization code.

3. The requesting site makes a HTTPS request to the OAuth AS to exchange the authorization code for an access token.

> **Note**
>
> If the OAuth client has provided the optional parameter `code_challenge` in step 1, it must submit the corresponding `code_verifier` in this request.

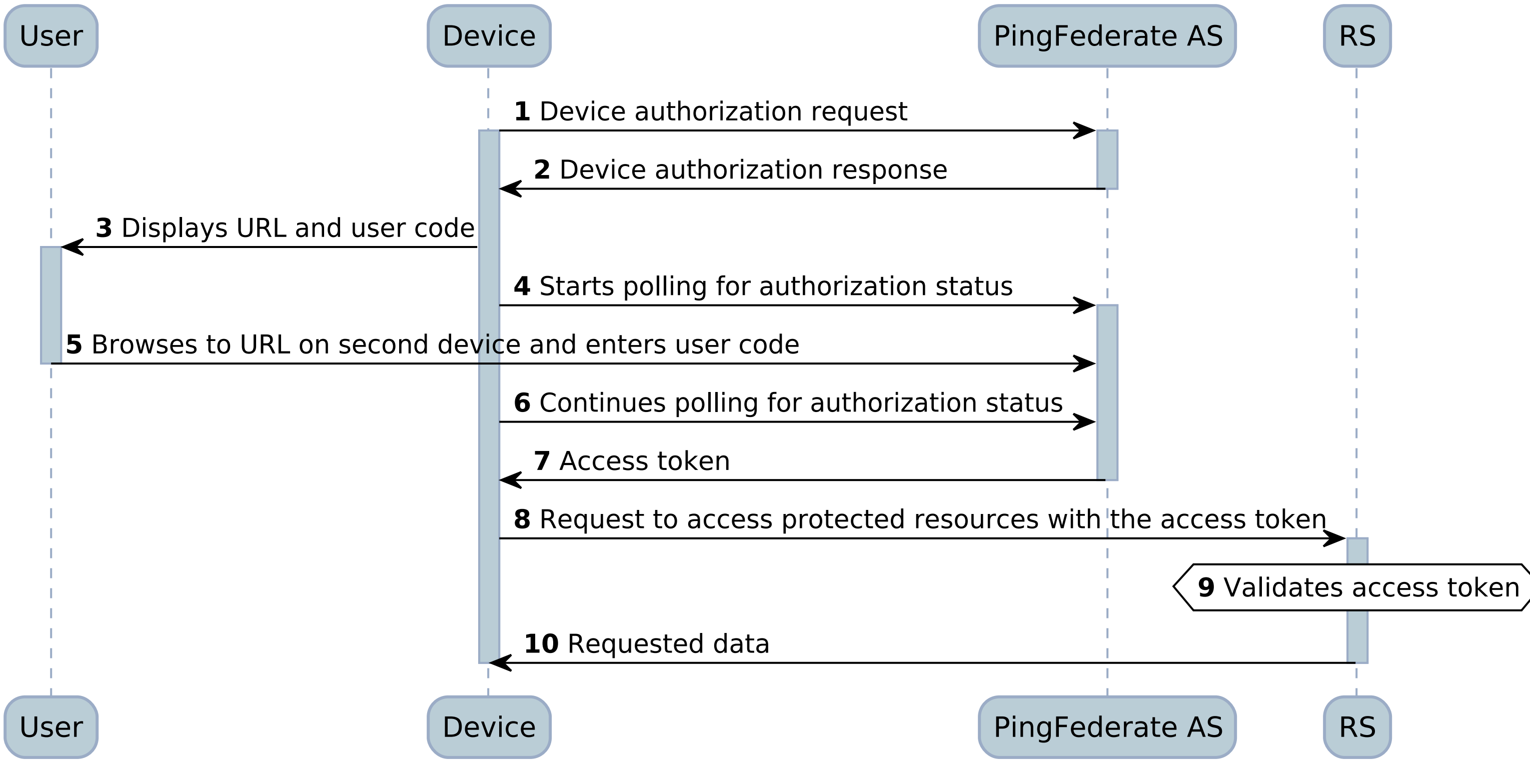
The OAuth AS validates the grant and user data associated with the code and then returns an access token.

4. The requesting site uses the access token in an API call to request user data.
5. The RS asks PingFederate for verification that the token is valid and has not expired. PingFederate returns data about the user, the granted scope, and the client ID.
6. Once verified, the RS returns the requested data to the requesting site.
7. The requesting site displays data from the API call to the user.

Device authorization grant

The device authorization grant process takes place between the user, device, PingFederate authorization server (AS), and resource server (RS).

In this scenario, a user attempts to access a protected resource through a device client that lacks a browser or has limited user-input capabilities, such as a smart TV, digital picture frame, or printer. The OAuth device authorization grant type allows a user to grant authorization to the device client using a browser on a second device, such as a smart phone or computer. For more information about the grant type, see the OAuth 2.0 Device Authorization Grant specification.


User
Device
PingFederate AS
RS
1 Device authorization request
2 Device authorization response
3 Displays URL and user code
4 Starts polling for authorization status
5 Browses to URL on second device and enters user code
6 Continues polling for authorization status
7 Access token
8 Request to access protected resources with the access token
9 Validates access token
10 Requested data
User
Device
PingFederate AS
RS

OAuth device authorization grant

Processing steps

1. The device sends a device authorization request to PingFederate, the authorization server (AS), at its device authorization endpoint.
2. PingFederate returns a device authorization response. Among other parameters, the response contains a device code, a user code, a user authorization endpoint, and a user authorization endpoint with the user code in a query parameter.
3. The device provides the user authorization endpoint with the user code in a query parameter, the user code, and instructions to the user, as in the following example.

   ```
   Using a browser on another device, visit:
   https://www.example.com/authorizeDevice

   Enter the code:
   HVF7-B4KW
   ```

4. The device starts sending device access token requests to PingFederate at its token endpoint to poll whether the user has completed the authorization process.

   The device access token request must include the device authorization grant type, `urn:ietf:params:oauth:grant-type:device_code`, the device code, and the user code.

   For each device access token request it receives, PingFederate returns a device access token response. The payload varies depending on the authorization status.
5. The user completes the authorization process by performing the following actions:
   1. Goes to the user authorization endpoint on a second device that has a browser, such as a smartphone or a computer.
   2. Fulfills the authentication requirements.
   3. Enters the user code or confirms a pre-populated user code.
   4. Approves or denies the scope of permissions requested by the device.
6. The device continues polling PingFederate for an authorization status.
7. PingFederate validates the user code and provides the device with an access token in the device access token response.

   If the user denies the scope of permissions, PingFederate provides the device with a relevant error message in the device access token response.
8. The device provides the access token to the RS to access protected resources.
9. The RS validates the access token.
10. The RS provides the requested data to the device.

Related links

- Grant types
- Configuring authorization server settings

- Device authorization endpoint
- User authorization endpoint

CIBA grant

Client Initiated Backchannel Authentication (CIBA) is an extension to OpenID Connect that improves the end-user experience during authentication and authorization in a federated environment.

Like OpenID Connect, CIBA is an authentication flow, governing how clients are identified and granted access. With CIBA, user consent can be requested through an out-of-band flow. For example, when making an online purchase, CIBA improves user experience because the customer’s browser will not have to redirect to a financial institution for authorization. Instead, the customer receives a push notification from the financial institution’s mobile app to complete authorization. This allows the customer to avoid confusing browser redirects.

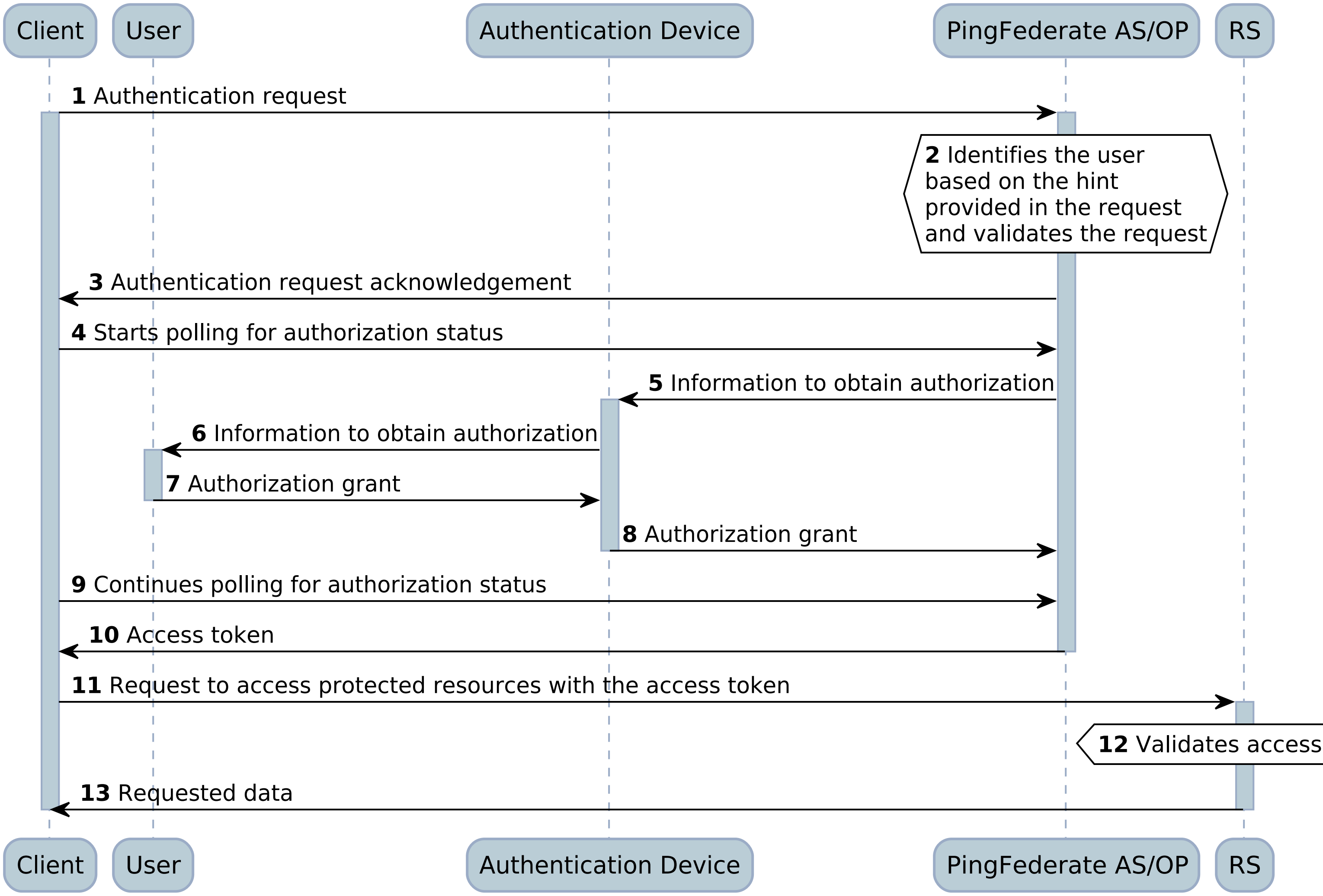
Related links

- Client Initiated Backchannel Authentication (CIBA)
- Client-initiated backchannel authentication endpoint

CIBA by poll

The OAuth Client Initiated Backchannel Authentication (CIBA) grant by poll process takes place between the client, user, authentication device, PingFederate, and resource server (RS).

After receiving an authentication request acknowledgment, the client starts polling the OpenID Provider (OP)'s token endpoint on a regular interval to obtain the authorization results. When the OP receives the authorization granted by the user through the authentication device, it returns an access token to the client.


Client
User
Authentication Device
PingFederate AS/OP
RS
1 Authentication request
2 Identifies the user based on the hint provided in the request and validates the request
3 Authentication request acknowledgement
4 Starts polling for authorization status
5 Information to obtain authorization
6 Information to obtain authorization
7 Authorization grant
8 Authorization grant
9 Continues polling for authorization status
10 Access token
11 Request to access protected resources with the access token
12 Validates access
13 Requested data
Client
User
Authentication Device
PingFederate AS/OP
RS

OAuth CIBA grant by poll

Processing steps

1. The client sends an authentication request to PingFederate at its client-initiated backchannel authentication endpoint. The client must include in its authentication request the desired scope of permissions and one identity hint for PingFederate to identify the user. When providing an identity hint, the client has three options:

   - `login_hint`
   - `login_hint_token`
   - `id_token_hint`

   The client can include a user code using the `user_code` parameter in the authentication request, transmit all request parameters of the authentication request in a signed request object, or do both.

2. PingFederate validates the authentication request and identifies the user based on the hint provided by the client.

3. PingFederate returns an authentication request acknowledgement to the client. The response contains the identifier, `auth_req_i`, that PingFederate assigns to the authentication request.
4. The client starts polling PingFederate at its token endpoint to check whether the user has completed the authorization process.

   The client must include in its token request the CIBA grant type, `urn:openid:params:grant-type:ciba`, and the corresponding `auth_req_id` value.

   For each token request it receives, PingFederate returns a token response. The payload varies depending on the authorization status.
5. PingFederate invokes a CIBA authenticator based on the applicable CIBA request policy to reach out to the user with the information (for example, the requested scopes) that the user needs to obtain authorization.
6. The authentication device presents the information and works with the user to obtain authorization.
7. The user reviews the information presented by the authentication device and then approves or denies the scopes requested by the client.
8. The authentication device sends the authorization result back to PingFederate.
9. The client continues polling PingFederate for an authorization result.
10. PingFederate returns an access token in a token response to the client.

    If the user denies the requested scopes, PingFederate provides the client with a relevant error message in the token response.
11. The client provides the access token to the RS to access protected resources.
12. The RS validates the access token.
13. The RS provides the requested data to the client.

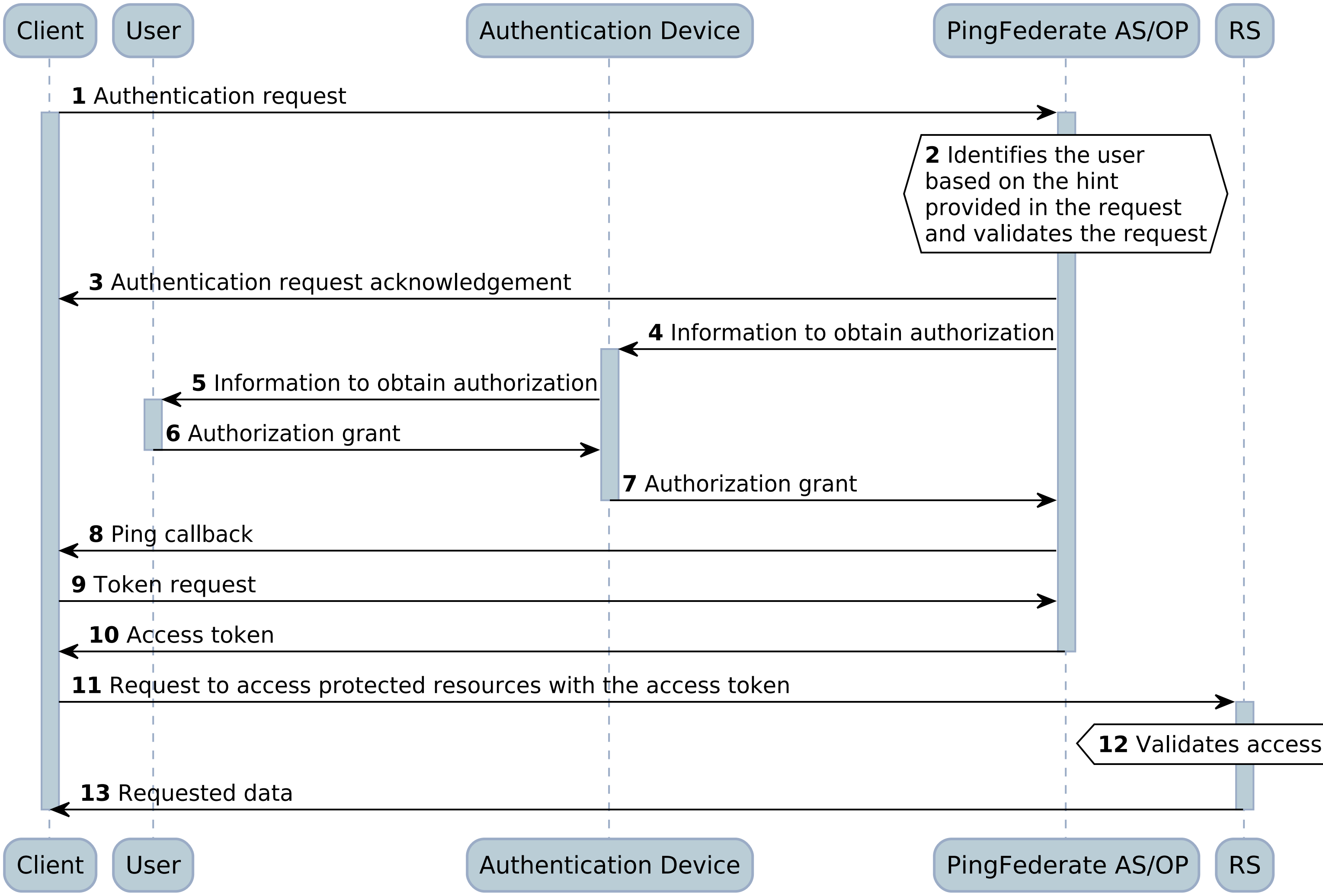
Related links

- OpenID Connect Client Initiated Backchannel Authentication Flow

CIBA by ping

The OAuth Client Initiated Backchannel Authentication (CIBA) grant by ping process takes place between the client, user, authentication device, PingFederate, and resource server (RS).

After receiving an authentication request acknowledgment, the client waits for a ping callback message from the OpenID Provider (OP). When the OP receives the authorization granted by the user through the authentication device, it sends a ping callback message to the client's notification endpoint. The client then sends a token request to retrieve an access token.


Client
User
Authentication Device
PingFederate AS/OP
RS
1 Authentication request
2 Identifies the user based on the hint provided in the request and validates the request
3 Authentication request acknowledgement
4 Information to obtain authorization
5 Information to obtain authorization
6 Authorization grant
7 Authorization grant
8 Ping callback
9 Token request
10 Access token
11 Request to access protected resources with the access token
12 Validates access
13 Requested data
Client
User
Authentication Device
PingFederate AS/OP
RS

OAuth CIBA grant by ping

Processing steps

1. The client sends an authentication request to PingFederate at its client-initiated backchannel authentication endpoint.

   The client must include in its authentication request the desired scope of permissions, one identity hint for PingFederate to identify the user, and a bearer token that PingFederate can use to authenticate the ping callback message. When providing an identity hint, the client has three options:

   - `login_hint`
   - `login_hint_token`
   - `id_token_hint`

   For the bearer token, the client must follow the syntax as defined in RFC 6750, section 2.1 and transmit it using the `client_notification_token` parameter.

The client can include a user code using the `user_code` parameter, transmit all request parameters of the authentication request in a signed request object, or do both.

The authentication request can be signed or unsigned.

2. PingFederate validates the authentication request and identifies the user based on the hint provided by the client.
3. PingFederate returns an authentication request acknowledgment to the client. The response contains the identifier, `auth_req_id`, that PingFederate assigns to the authentication request.
4. PingFederate invokes a CIBA authenticator based on the applicable CIBA request policy to reach out to the user with the information (for example, the requested scopes) that the user needs to obtain authorization.
5. The authentication device presents the information and works with the user to obtain authorization.
6. The user reviews the information presented by the authentication device and then approves or denies the scopes requested by the client.
7. The authentication device sends the authorization result back to PingFederate.
8. PingFederate sends a ping callback message using the HTTP POST method to the client at its notification endpoint.

   Per specification, PingFederate includes the `client_notification_token` value in the Authorization HTTP request header and the `auth_req_id` value in the message body.
9. The client sends a token request to PingFederate at its token endpoint.

   The client must include in its token request the CIBA grant type, `urn:openid:params:grant-type:ciba`, and the corresponding `auth_req_id` value.
10. PingFederate returns an access token in a token response to the client.

    If the user denies the requested scopes, PingFederate provides the client with a relevant error message in the token response.
11. The client provides the access token to the RS to access protected resources.
12. The RS validates the access token.
13. The RS provides the requested data to the client.

Related links

- OpenID Connect Client Initiated Backchannel Authentication Flow

Token exchange grant

You can configure the PingFederate OAuth server to support the token exchange grant type.

This feature uses the protocol defined in the OAuth 2.0 token exchange specification RFC 8693. Learn more about OAuth 2.0 token exchange in RFC 8693 specifications. For information about configuring the OAuth server for token exchange, see OAuth token exchange.

PingFederate supports the following token types:

- urn:ietf:params:oauth:token-type:access_token

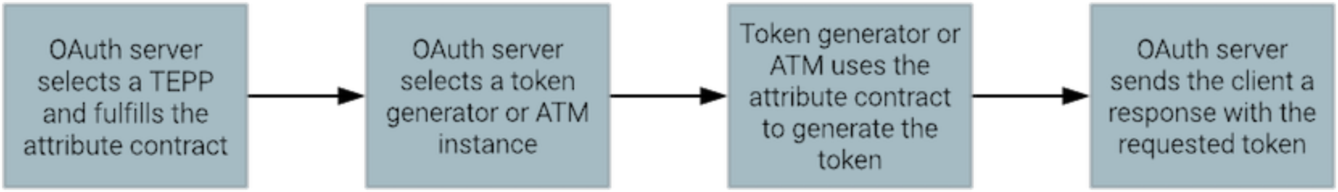
- urn:ietf:params:oauth:token-type:saml1
- urn:ietf:params:oauth:token-type:saml2
- urn:params:oauth:token-type:jwt

An OAuth token exchange begins when an OAuth client sends a token exchange request to the PingFederate OAuth server. The request contains the token exchange grant type parameter, a subject token, and a subject token type parameter. For impersonation use cases, the request also contains an actor token and actor token type parameter. The request might also contain resource, audience, scope, and requested token type parameters.

The optional resource parameter identifies an Access Token Management (ATM) instance or token generator instance. The optional audience parameter identifies another client that will use the new token to get access to a resource. If a request includes an audience parameter, the OAuth server will select the ATM instance specified in the audience client's configuration as seen in step 2.

A successful token exchange ends when the OAuth server sends the client a message containing the requested token. The response also contains an issued token type parameter and a token type parameter.

To function as the authorization server, PingFederate requires at least one Token Exchange Processor Policy (TEPP) to use the token exchange grant type. Learn more in Defining token exchange processor policies. A TEPP includes a map of subject token types to token processor instances and a map of attributes from the request and other sources to a TEPP attribute contract. You must assign a default TEPP to the OAuth server. Optionally, you can assign a TEPP to OAuth clients that need to exchange tokens.


OAuth server selects a TEPP and fulfills the attribute contract
OAuth server selects a token generator or ATM instance
Token generator or ATM uses the attribute contract to generate the token
OAuth server sends the client a response with the requested token

When the OAuth server receives a token exchange request from an OAuth client:

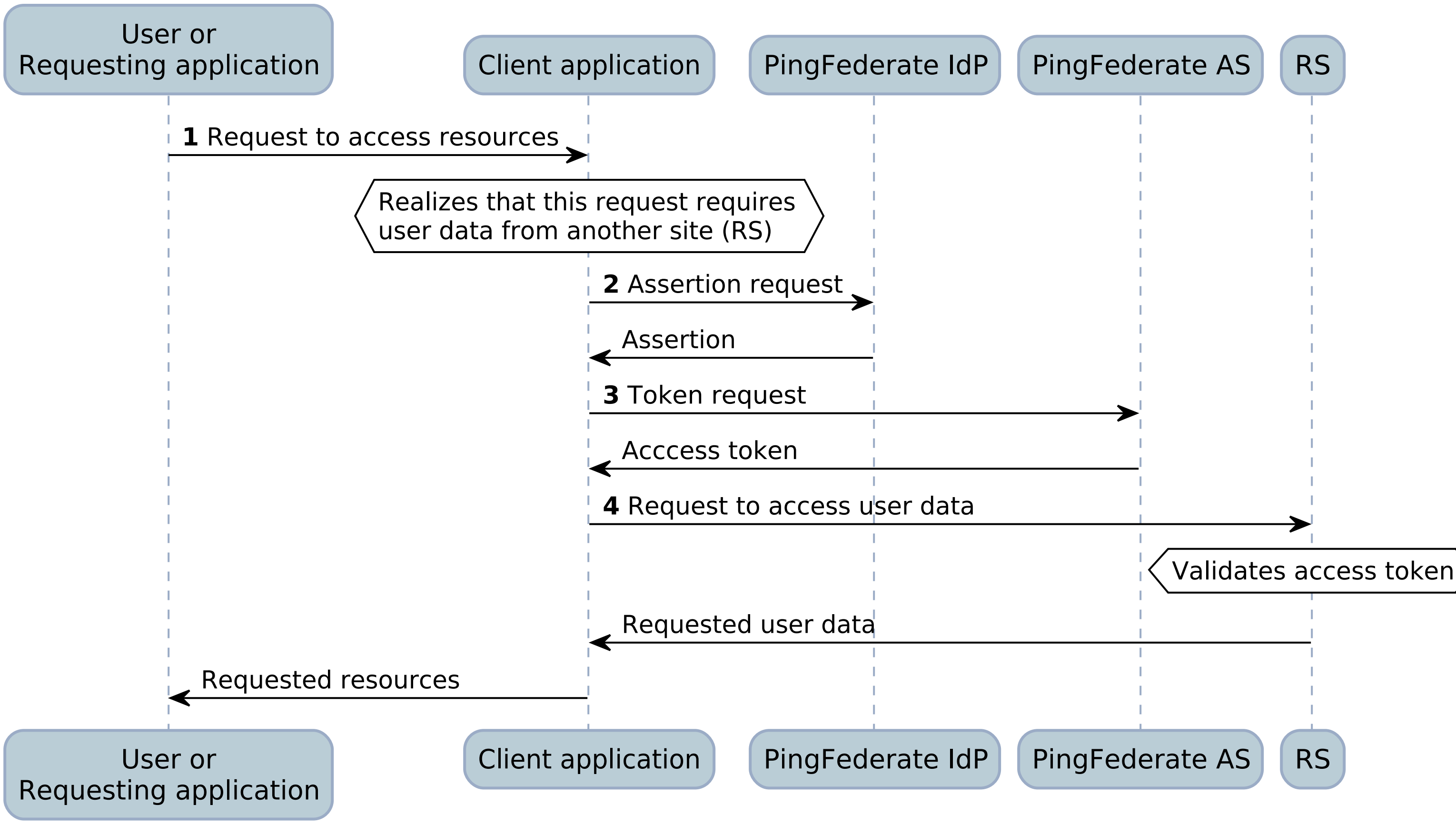
1. The OAuth server selects a TEPP and fulfills the TEPP attribute contract as follows:
   - If the OAuth client's configuration specifies a TEPP, the OAuth server selects that TEPP. Otherwise, the OAuth server selects the default OAuth server TEPP.
   - The OAuth server uses the TEPP's map of subject token types to select a token processor instance. The token processor then uses the TEPP's attribute map to produce a TEPP attribute contract.
2. The OAuth server selects a token generator instance or ATM instance as follows:
   - If the request does not have an audience or resource parameter, the OAuth server selects the default OAuth server ATM instance.
   - If the request includes resource parameters, audience parameters, or both, and if the OAuth server determines that they identify a single ATM or token generator instance, the OAuth server selects it.
   - If the request includes resource parameters, audience parameters, or both, and if the OAuth server determines that they don't identify a single ATM or token generator instance, the OAuth server returns an error.

3. The token generator or ATM uses the attribute contract to generate the token as follows:
    - The default token type is generated if the request doesn't include a requested token type parameter.
    - If the request includes a requested token type parameter, that type of token is generated.
    - If the request includes a requested token type but the token generator or ATM doesn't support that token type, the OAuth server returns an error.
4. The OAuth server sends the client a response containing the requested token.

Assertion grant profile for OAuth 2.0 authorization grants

The assertion grant profile process takes place between the user or requesting application. client application, PingFederate identity provider (IdP), PingFederate authorization server (AS) and resource server (RS).

In this scenario, a client obtains an assertion, either a SAML 2.0 bearer assertion or a JSON Web Token (JWT) bearer token, and makes an HTTP request to the PingFederate OAuth AS to exchange the assertion for an access token. The OAuth AS validates the assertion and returns an access token. The client uses the token in an API call to the RS to obtain data.



Assertion grant profile

Processing steps

1. A user-initiated or client-initiated event, such as a mobile application or a scheduled task, requests access to software as a service (SaaS) protected resources from an OAuth client application.

2. The client application obtains an assertion from an IdP.

> **Note**
>
> When using SAML assertions as authorization grants, client applications must obtain assertions that meet the requirements defined in RFC7522. Do not use SAML assertions acquired through browser single sign-on (SSO) profiles here.

3. The client application makes an HTTP request to the PingFederate OAuth AS to exchange the assertion for an access token. The OAuth AS validates the assertion and returns the access token.

4. The client application adds the access token to its API call to the RS. The RS returns the requested data to the client application.

Related links

- JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants
- Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants
- Configuring an OAuth assertion grant IdP connection

OpenID Connect support

As an extension of OAuth capabilities, PingFederate supports an optional configuration for OpenID Connect.

OpenID Connect is a modern protocol for secure, lightweight transfer of authentication and user attributes. For more information, see openid.net/connect.

PingFederate can be deployed as an OpenID Provider (OP), a Relying Party (RP), or both. PingFederate supports both the Basic Client and the Implicit Client profiles.

Client management

PingFederate provides features that support OAuth client management.

PingFederate provides administrators the flexibility to manage OAuth clients using the following interfaces:

- The administrative console
- The administrative API
- The OAuth Client Management Service

Additionally, PingFederate supports dynamic client registration based on the OAuth 2.0 Dynamic Client Registration Protocol specification.

System for Cross-domain Identity Management (SCIM)

PingFederate supports the SCIM 1.1 protocol for outbound and inbound provisioning.

At an identity provider (IdP) outbound site, you have the option to automatically provision and maintain user accounts at service provider (SP) sites that have implemented SCIM. When you have PingFederate configured as an SP inbound site, you can automatically provision and manage user accounts and groups for your own organization using the standard SCIM protocol. For a brief summary of supported features, see the following table.

Feature	Outbound provisioning	Inbound provisioning
SCIM specification	SCIM 1.1	SCIM 1.1
Data format	JSON	JSON
User and group create, read, update, and delete (CRUD) operations	Yes	Yes
Custom schema support	Yes	Yes
List/query and filtering support	Not applicable	Yes
PATCH	Yes	No
Authentication method	HTTP Basic and OAuth Resource Owner Password Credentials grant type	HTTP Basic and client certificate (mutual TLS)
Source data stores	PingDirectory, Microsoft Active Directory, and Oracle Unified Directory	Not applicable
Target data stores	Not applicable	Active Directory and other data stores via the Identity Store Provisioner Java SDK interface

For detailed information about SCIM, see www.simplecloud.info.

Transport and message security

Two main ways of securing interactions are Secure Sockets Layer with Transport Level Security (SSL/TLS) and digital signatures.

Use SSL/TLS in environments the require both message confidentiality and integrity. For SAML messaging, digital signatures ensure the identity of both parties involved in the transaction and validate that a particular partner received a message. To achieve increased privacy, PingFederate also lets you encrypt SAML 2.0 messages, including SAML metadata files, as well as WS-Trust STS assertions.

For more information, see Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0.

Integration overview

As a standalone server, PingFederate must be programmatically integrated with end-user applications and identity management (IdM) systems to complete the “first- and last-mile” implementation of a federated identity network for browser-based single-sign-on (SSO).

PingFederate supports identity and access management (IAM) integrations with a wide range of cloud, mobile, Software as a Service (SaaS), APIs, and on-premises applications.

For an identity provider (IdP), the first mile of this integration process involves providing a mechanism through which PingFederate looks up a user’s current authenticated session data, such as a cookie, or authenticates a user without such a session. For a service provider (SP), the last mile involves enabling PingFederate to supply information needed by the target application to set a valid session cookie or other application-specific security context for the user.

To enable both sides of this integration, PingFederate provides bundled and separately available integration kits, which include adapters that plug into the PingFederate server and agent toolkits that interface with local IdM systems or applications, as needed.

In addition, PingFederate provides plugin authentication selectors, which enable dynamic selection of authentication sources based on administrator-specified criteria.

For integrations that are included with PingFederate, see Bundled adapters and authenticators. For an overview of the various types of integrations that are available, see Additional integrations.

Visit the Ping Identity Integration Directory to explore 1800+ integrations and find links to downloads and documentation.

Bundled adapters and authenticators

PingFederate comes bundled with the following adapters and authenticators to enable common deployment scenarios.

Bundled adapters

Composite Adapter

Allows multiple configured identity provider (IdP) adapters to execute in sequence. Depending on the authentication context, use this capability, called adapter chaining, for either single-adapter usage or to support multi-factor authentication through a series of adapters. Learn more in Composite Adapter.

HTML Form Adapter

Used in conjunction with Password Credential Validators. These adapters provide integration with user-datastores in directory servers or locally. Learn more in HTML Form Adapter.

HTTP Basic Adapter

Used in conjunction with Password Credential Validators. These adapters provide integration with user-data stores in directory servers or locally. Learn more in HTTP Basic Adapter.

Identifier First Adapter

When a variety of user types authenticate at PingFederate, it is often better to ask the user for their identifier first, determine their user population, and prompt the user with the desired authentication requirements and experience. The Identifier First Adapter is designed to handle this use case.

Kerberos Adapter

Provides a seamless desktop SSO experience for Windows environments and supports authentication mechanism assurance from the Active Directory domain service. For new configurations and as a simpler alternative to the separately-available IWA Integration Kit, use this adapter. Learn more in Kerberos Adapter.

OpenToken Adapter

Provides a generic interface for integrating with various applications, including Java- and .NET-based applications. Learn more in OpenToken Adapter.

Passthrough IdP Adapter

The Passthrough IdP Adapter allows a user key to be associated with a user's authentication sessions. By placing the Passthrough IdP Adapter downstream from an IdP connection in a policy tree, you can take advantage of additional capabilities associated with defining a user key. Learn more in Configuring a Passthrough IdP Adapter.

PingID Adapter

PingID is a cloud-based authentication service that binds user identities to their devices, making it an effective multi-factor authentication solution. Learn more in the PingID documentation.

PingOne DaVinci Adapter

Allows PingFederate to use PingOne as an IdP as part of your PingFederate authentication policy. You can find detailed information in the PingOne DaVinci Integration Kit.

PingOne MFA Adapter

Allows PingFederate to use the PingOne MFA service for multi-factor authentication (MFA). You can find detailed information in the PingOne MFA Integration Kit.

PingOne Protect Adapter

When a user signs on through PingFederate, the adapter sends the transaction information to the PingOne Protect service and retrieves a risk evaluation and other information about the user's current and previous transactions. You can find detailed information in the PingOne Protect Integration Kit.

PingOne Verify Adapter

Allows PingFederate to use the PingOne Verify service to trigger an identity verification challenge as part of the PingFederate authentication policy or registration flow. For example, you can use this adapter for personal identity verification based on a government issued photo ID. You can find detailed information in the PingOne Verify Integration Kit.

Bundled authentication selectors

PingFederate provides plugin authentication selectors, which enable dynamic selection of authentication sources based on administrator-specified criteria. Along with the Composite Adapter and token authorization, the selectors enable dynamic integration with an organization’s authentication or authorization policies, also known as adaptive federation.

> **Tip**
>
> To select subsequent selectors or authentication sources for handling complex hierarchical access-policy decisions, use the results of authentication-selection criteria evaluation. Learn more in Authentication policies.

CIDR Authentication Selector

Provides a means of choosing authentication sources or other authentication sources at runtime based on whether an end-user’s IP address falls within specified ranges using Classless Inter-Domain Routing notation. This selector allows administrators to determine, for example, whether an SSO request originates inside or outside the corporate firewall and use different authentication integration accordingly. Learn more in Configuring the CIDR Authentication Selector.

Cluster Node Authentication Selector

Provides a means of picking authentication sources or other authentication sources at runtime based on the PingFederate cluster node that is servicing the request. For example, you can configure this selector to choose whether PingFederate attempts Integrated Windows Authentication based on the PingFederate cluster node with which a Key Distribution Center is associated. Learn more in Configuring the Cluster Node Authentication Selector.

Connection Set Authentication Selector

Provides a means of selecting authentication sources or other authentication sources at runtime based on a match found between the target SP connection used in an SSO request and SP connections configured within PingFederate. For example, administrators with different requirements for SP connections can override connection adapter selection on an individual connection basis. Learn more in Configuring the Connection Set Authentication Selector.

Extended Property Authentication Selector

Enables PingFederate to choose configured authentication sources or other selectors based on a match found between a selector result value and an extended property value from the invoking browser-based SSO connections or OAuth client. Learn more in Configuring the Extended Property Authentication Selector.

HTTP Header Authentication Selector

Provides a means of choosing authentication sources or other authentication sources at runtime based on a match found using wildcard expressions in an HTTP header. This selector allows administrators to determine, for example, authentication behavior based on the type of browser. Learn more in Configuring the HTTP Header Authentication Selector.

HTTP Request Parameter Authentication Selector

Provides a means of selecting authentication sources or other authentication sources at runtime based on query parameter values in the HTTP request. Learn more in Configuring the HTTP Request Parameter Authentication Selector.

OAuth Client Set Authentication Selector

Enables PingFederate to choose configured authentication sources or other selectors based on a match found between the client information in an OAuth request and the OAuth clients configured in the PingFederate OAuth authorization server (AS). This selector allows you to override client authentication selection on an individual client basis in one or more authentication policies. Learn more in Configuring the OAuth Client Set Authentication Selector.

OAuth Scope Authentication Selector

Provides a means of selecting authentication sources or other authentication sources at runtime based on a match found between the scopes of an OAuth authorization request and scopes configured in the PingFederate OAuth authorization server (AS). For example, if a client requires write access to a resource, administrators can configure the selector to choose an adapter that offers a stronger form of authentication such as the X.509 client certificate rather than username and password. Learn more in Configuring the OAuth Scope Authentication Selector.

Requested AuthN Context Authentication Selector

Provides a means of picking authentication sources or other authentication sources at runtime based on the authentication context requested by an SP, for SP-initiated SSO. Configured authentication sources are mapped either to SAML-specified contexts or any ad-hoc context agreed upon between the IdP and SP partners. Learn more in Configuring the Requested AuthN Context Authentication Selector.

Session Authentication Selector

Enables PingFederate to choose a policy path at runtime based on whether the user already has a PingFederate authentication session for a particular source. Learn more in Configuring the Session Authentication Selector.

> **Note**
>
> Authentication selectors rely on HTTP requests, HTTP headers, POST data, or a combination of these authentication sources. Ensure that standard security measures are in place when using these selectors.

Software development kit (SDK)

The PingFederate SDK provides a flexible means of creating custom adapters to integrate federated identity management into your system environment. Learn more in the PingFederate SDK Developer’s Guide.

Additional integrations

In addition to the bundle integrations and authenticators, Ping Identity provides identity and access management (IAM) integrations with a wide range of cloud, mobile, software as a service (SaaS), APIs, and on-premises applications.

To discover integrations, including downloads and documentation, visit the Ping Identity Integration Directory.

The following describes our most common types of integrations:

Single sign-on (SSO)

Integration kits allow PingFederate to coordinate SSO for a variety of platforms and applications, such as Atlassian and Citrix.

For select platforms, configuration guides are provided to help you configure SSO without any additional downloads.

User and group provisioning

Connectors allow PingFederate to provision and synchronize users and groups from an attached datastore to a directory in a cloud service, such as PingOne, Salesforce, or Zoom.

Connectors also let you set up optional SSO between PingFederate and the cloud service.

These typically include a connection template for the PingFederate provisioning engine.

Multi-factor authentication (MFA)

MFA integrations allow PingFederate to include third-party MFA providers, such as PingOne MFA and RSA SecurID, as part of the sign-on flow.

Mobile device management

Mobile device management integrations allow PingFederate to adjust the sign-on flow based on device information from services such as MobileIron and AirWatch.

These typically include an IdP adapter.

Risk-based intelligent authentication

Risk intelligence integration kits allows PingFederate to retrieve a security risk assessment from third parties, such as PingOne Protect and ThreatMetrix, when a user signs on. You can use this information to dynamically adjust authentication requirements based on the risk level for each sign-on event.

These typically include an IdP adapter. For more information, see PingOne Protect Integration Kit.

Custom application integration

The Agentless Integration Kit can integrate any application with PingFederate for SSO. This uses API calls with no need for agent software. We provide sample code for a variety of programming languages.

Specific integrations are available for the language your application uses, such as Java, .NET, and PHP. These include an IdP and SP adapter, an agent file, and sample code.

Web server agents

PingFederate to extend single sign-on abilities to applications running on web servers, such as Apache and IIS.

Social sign-on

Cloud identity connectors (CIC) allow PingFederate to coordinate SSO by using third-party services as identity providers (IdP). This allows a user to access a service provider (SP), such your web application, by signing into a popular service, such as LinkedIn, Google, or Facebook.

These typically include a PingFederate IdP adapter.

Checking user credentials

Password credential validator (PCV) integrations let PingFederate validate user credentials against a directory service, such as PingOne and Azure, when a user signs on.

These include a PCV add-on for PingFederate.

Converting identity tokens

Token translators allow PingFederate to convert or validate IdP tokens in a variety of formats, such as the WS-Trust security token service (STS).

These typically include token processor and token generator files.

Custom integrations

PingFederate also includes an SDK that software you can use to write custom interfaces for specific systems. For more information, see the SDK Developer’s Guide.

Specific use cases

To explore comprehensive guides for specific use cases, see Use Case Guides.

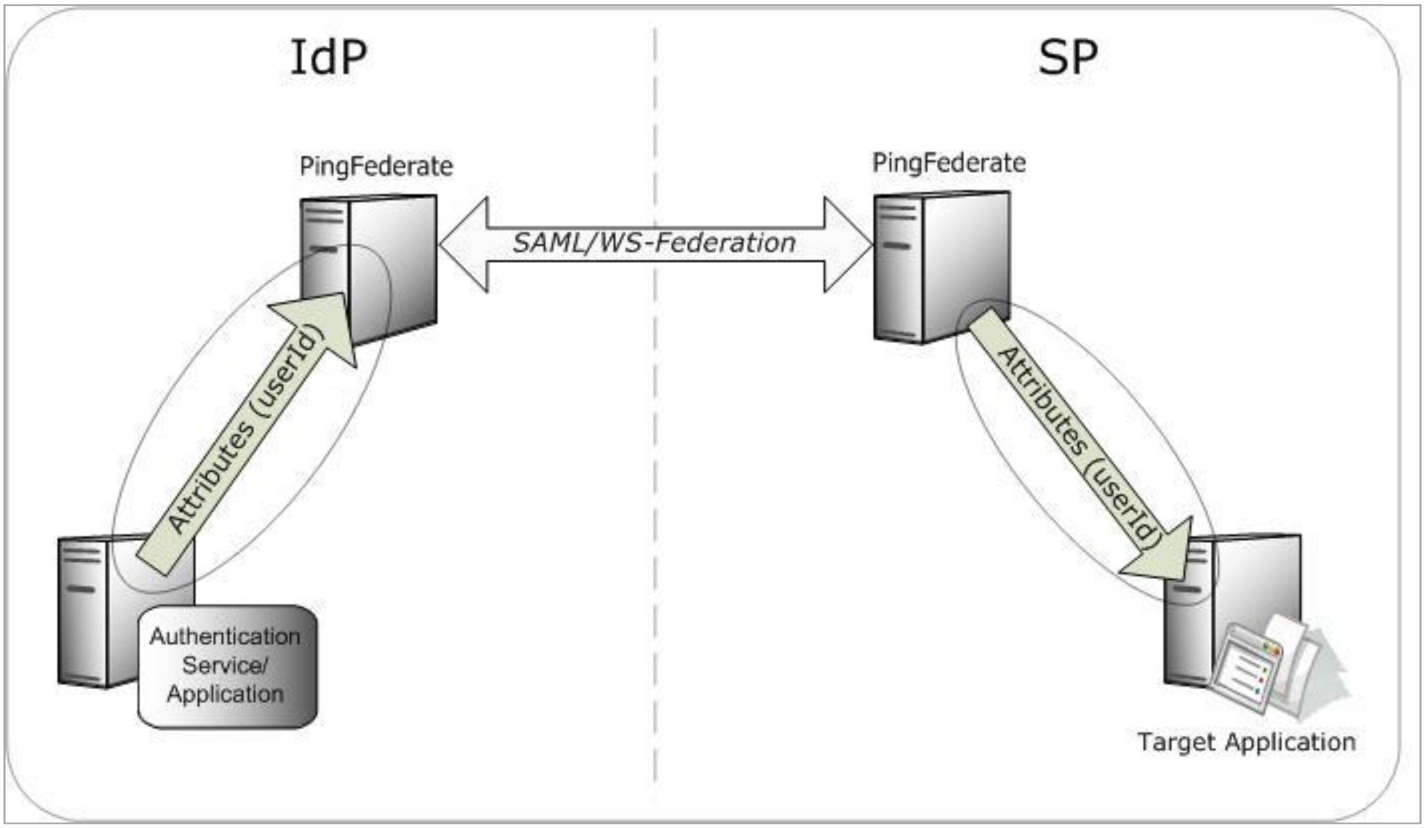
SSO integration concepts

PingFederate supports both identity provider (IdP) and service provider (SP) integration.

For an IdP, the first step in the integration process involves sending identity attributes from an authentication service or application to PingFederate. PingFederate uses those identity attributes to generate a SAML assertion. For information about SAML, see Supported standards. IdP integration typically provides a mechanism through which PingFederate looks up a user’s current authenticated session data, such as a cookie, or authenticate a user without such a session.

For an SP, the last step of the integration process involves sending identity attributes from PingFederate to the target application. PingFederate extracts the identity attributes from the incoming SAML assertion and sends them to the target application to set a valid session cookie or other application-specific security context for the user.

The following diagram illustrates the basic concepts of integration with PingFederate.

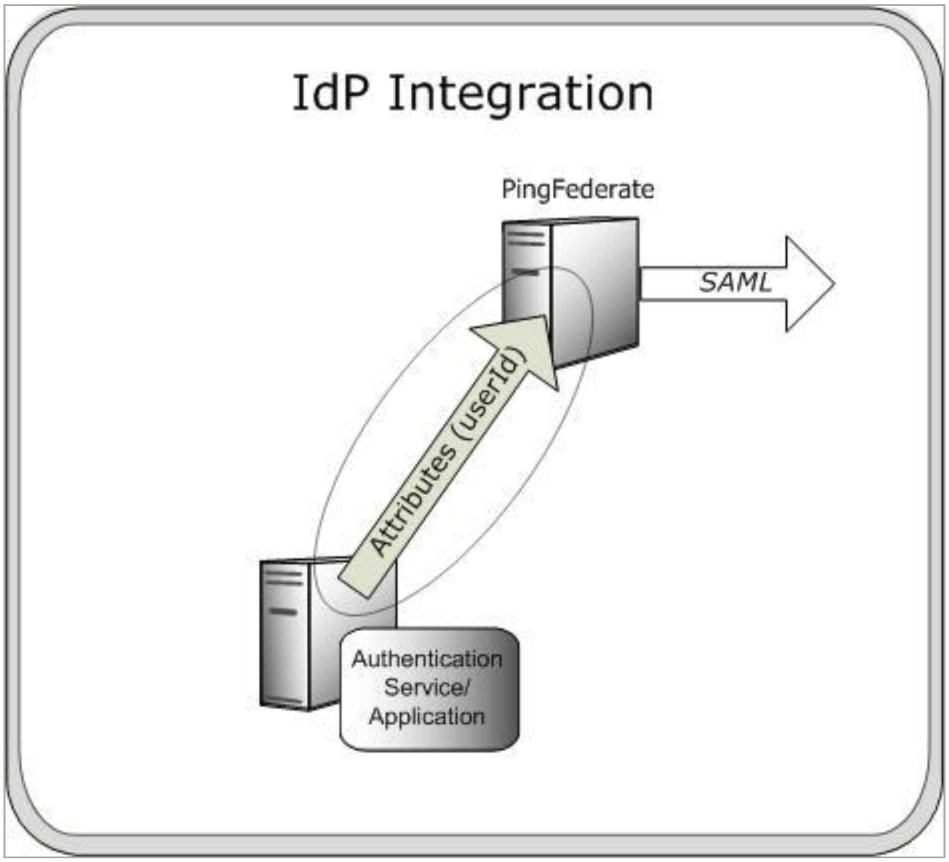


Identity provider integration

Identity provider (IdP) integration involves retrieving user-identity attributes from the IdP domain and sending them to the PingFederate server.

An IdP is a system entity that authenticates a user, or “SAML subject,” and transmits referential identity attributes based on that authentication to PingFederate. Typically, the IdP retrieves the identity attributes from an authenticated user session. Depending on the IdP deployment/implementation enviornment, a number of attribute-retrieval approaches are used for IdP integration. Ping Identity offers a broad range of commercial integration kits that address various IdP scenarios, such as custom-application integration, integration with a commercial identity management (IdM) product, or integration with an authentication system.

> **Tip**
>
> For IdPs implementing single sign-on (SSO) to selected software as a service (SaaS) providers such as Google Apps and Salesforce, PingFederate also provides automated user provisioning.



Custom applications

Many applications use their own authentication mechanisms, typically through a database or LDAP repository, and are responsible for their own user-session management. Custom-application integration is necessary when there is limited or no access to the web or application server hosting the application. Integration with these custom applications is handled by application-level integration kits, which allow software developers to integrate their applications with a PingFederate server acting as a service provider (SP).

With these integration kits, PingFederate sends the identity attributes from the SAML assertion to the SP application, which then uses them for its own authentication and session management. As for the IdP, application-specific integration kits include an SP agent, which resides with the SP application and provides a simple programming interface to extract the identity attributes sent from the PingFederate server. PingFederate cam use this information to start a session for the SP application.

Ping Identity provides custom-application integration kits for a variety of programming environments, including:

- Java
- .NET
- PHP

Ping Identity provides an Agentless Integration Kit, which allows developers to use direct HTTP calls to the PingFederate server to temporarily store and retrieve user attributes securely, eliminating the need for an agent interface.

IdM systems

An IdP enterprise that uses an IdM system expands the reach of the IdM domain to external partner applications through integration with PingFederate. IdM integration kits typically use the IdM agent API to access identity attributes in the IdM proprietary session cookie and transmit those attributes to the PingFederate server.

IdM integration kits do not require any development; the PingFederate administrative console accomplishes all integrations.

Ping Identity provides integration kits for many of the leading IdM systems, such as Oracle Access Manager.

Authentication systems

An authentication application or service normally handles initial user authentication outside of the PingFederate server. To access applications outside the security domain, PingFederate authentication-system integration kits leverage this local authentication.

Authentication integration kits do not require any development; the PingFederate administrative console accomplishes all integrations with PingFederate. Ping Identity offers integration kits for authentication systems including:

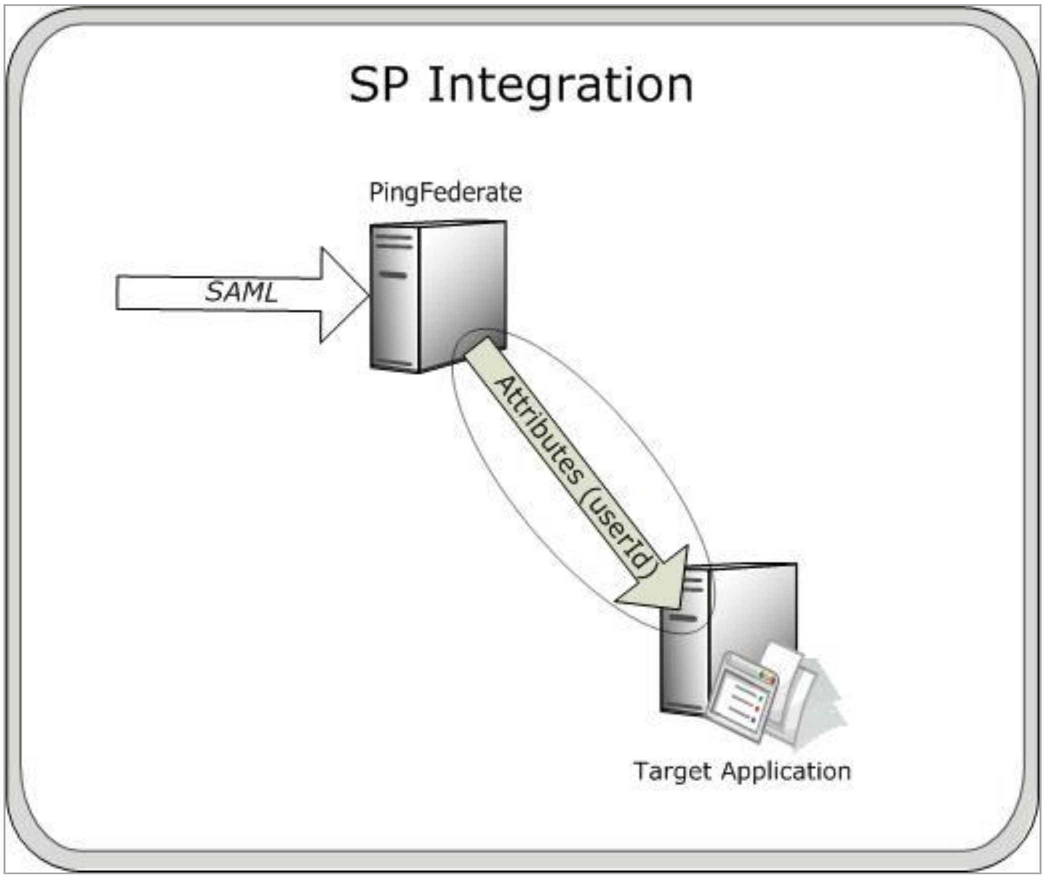
- X.509 Certificate
- RSA SecurID Integration Kit
- Symantec VIP Integration Kit

PingFederate also packages two IdP adapters, an HTML Form Adapter and an HTTP Basic Adapter, which delegate user authentication to plugin password credential validators (PCVs). Supplied validators use either an LDAP directory, RADIUS server, or a simple username/password verification system maintained by PingFederate. Customized validators can also be developed. When the PingFederate IdP server receives an authentication request for SP-initiated SSO or a user clicks a link for IdP-initiated SSO, PingFederate invokes the implemented adapter and prompts the user for credentials, if the user is not already logged on.

Service provider integration

Service provider (SP) integration involves passing the identity attributes from PingFederate to the target SP application.

An SP is the consumer of identity attributes provided by the identity provider (IdP) through a SAML assertion. The SP application uses this information to set a valid session or other security context for the user, represented by the identity attributes. Session creation involves a number of approaches. For the IdP, Ping Identity offers commercial integration kits that address the various SP scenarios. Most SP scenarios involve custom-application integration, server-agent integration, integration with an identity management (IdM) product, or integration with a commercial application.



Custom applications

Many applications use their own authentication mechanisms, typically through a database or LDAP repository, and are responsible for their own user-session management. Custom-application integration is necessary when there is limited or no access to the web or application server hosting the application. Application-level integration kits handle integration with these custom applications and allow software developers to integrate their applications with a PingFederate server acting as an SP.

With these integration kits, PingFederate sends the identity attributes from the SAML assertion to the SP application, which can then use them for its own authentication and session management. As for the IdP, application-specific integration kits include an SP agent, which resides with the SP application and provides a simple programming interface to extract the identity attributes sent from the PingFederate server. PingFederate cam use this information to start a session for the SP application.

Ping Identity provides custom-application integration kits for a variety of programming environments, including:

- Java
- .NET
- PHP

In addition, Ping Identity provides an Agentless Integration Kit, which allows developers to use direct HTTP calls to the PingFederate server to temporarily store and retrieve user attributes securely, eliminating the need for an agent interface.

Server agents

Server-agent integration with PingFederate allows SP enterprises to accept SAML assertions and provide single sign-on (SSO) to all applications running on that web or application server. You don't need to integrate each application. Because integration occurs at the server level, server-agent integration maximizes ease of deployment and scalability. Applications running on the web or application server must delegate authentication to the server. If the application employs its own authentication mechanism, integration must occur at the application level.

With server-agent integration kits, PingFederate sends the identity attributes from the SAML assertion to the server agent, which is typically a web filter or Java Authentication and Authorization Service (JAAS) Login Module. The server agent extracts the identity attributes, which the server then uses to authenticate and create a session for the user.

SP server-integration kits don't require any development work. The PingFederate administrative console accomplishes all integrations with PingFederate.

Ping Identity provides integration kits for many web and application servers, including:

- Internet Information Services (IIS)
- Apache (Red Hat)
- Apache (Windows)
- NetWeaver

IdM systems

IdM integration with PingFederate allows an SP enterprise to accept SAML assertions and provide SSO to applications protected by the IdM domain. IdM integration kits typically use the IdM agent API to create an IdM proprietary session token based on the identity attributes received from PingFederate.

IdM integration kits do not require any development; the PingFederate administrative console and the IdM administration tool accomplish integration with PingFederate.

Ping Identity provides integration kits for leading IdM systems, such as Oracle Access Manager.

Commercial applications and SaaS

Commercial-application integration with PingFederate allows an SP enterprise to accept SAML assertions and provide SSO to those commercial applications.

These integration kits don't require any development. The PingFederate administrative console accomplishes all integrations.

Ping Identity offers integration kits to many commercial applications and SaaS vendors, including:

- Citrix
- SharePoint
- Box
- Google
- Office 365
- Salesforce

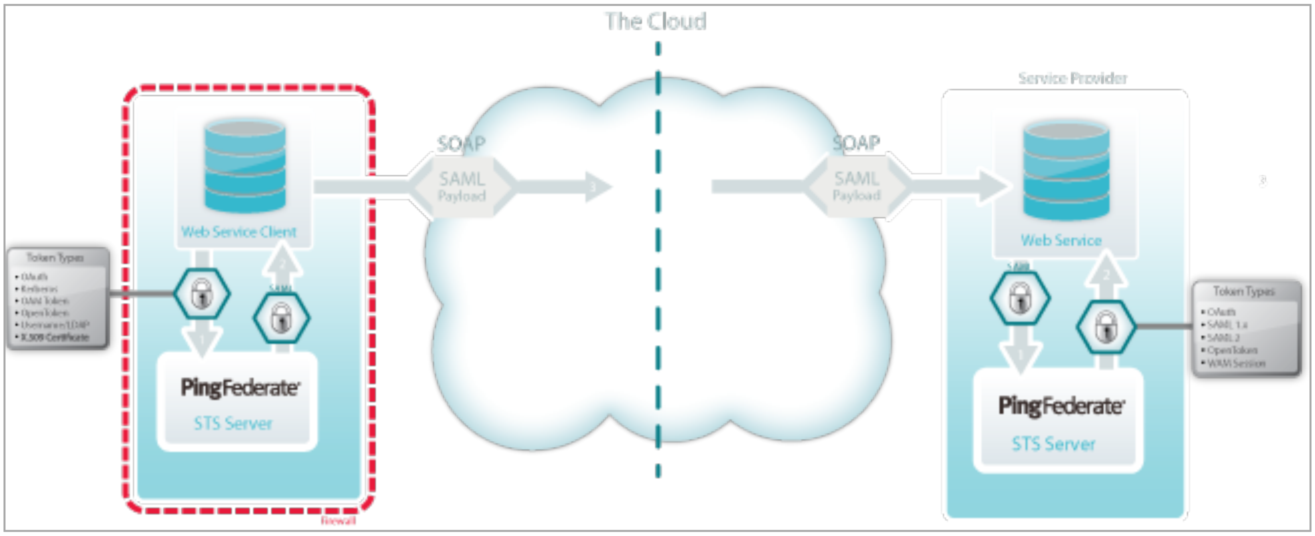
- Slack
- Workday
- Zendesk

Security token service

The PingFederate WS-Trust Security Token Service (STS) allows organizations to extend single sign-on (SSO) identity management (IdM) to web services.

The STS shares the core functionality of PingFederate, including console administration, identity and attribute mapping, and certificate security management. With PingFederate, web services identify the end user who has initiated a transaction across domains, providing enhanced service while simultaneously ensuring appropriate information access and regulatory accountability. For information about WS-Trust and the role of an STS, see Web services standards.

You can use PingFederate in many different scenarios to address different identity and security problems as they relate to web services, service-oriented architecture (SOA), and Enterprise Service Buses. All of these scenarios share a recommended architectural approach that uses a SAML assertion as the standard security token shared between security domains. For more information, see WS-Trust STS.



WS-Trust Security Token Service SSO

OAuth authorization server

PingFederate can act as an OAuth authorization server (AS), allowing a resource owner to grant authorization to a client requesting access to resources protected by a resource server (RS).

The OAuth AS issues tokens to clients on behalf of a resource for use in authenticating a subsequent API call—typically, but not exclusively a REST API. The PingFederate OAuth AS issues tokens to clients in several different scenarios, including:

- A web application wants access to a protected resource associated with a user and needs the user's consent.
- A native application client on a mobile device or tablet wants to connect to a user's online account and needs the user's consent.

- An enterprise application client wants to access a protected resource hosted by a business partner, customer, or software as a service (SaaS) provider.

For information about OAuth and the role of an AS, see OAuth 2.0 and PingFederate AS.

You can configure the PingFederate OAuth AS independently or in conjunction with security token service (STS) and browser-based single sign-on (SSO) for either an identity provider (IdP) or a service provider (SP) deployment. For more information, see About OAuth.

> **Note**
>
> OAuth AS capabilities might require additional licenses. For more information, contact sales@pingidentity.com.

User account management

Typically, the identity provider (IdP) repository maintains user accounts in an identity federation. However, a service provider (SP) often has its own set of user accounts, which might not always correspond to IdP users.

The SP might need to establish and maintain parallel accounts for remote single sign-on (SSO) users to enforce authorization policy, customize user experience, comply with regulations, or a combination of such purposes.

PingFederate provides two kinds of user provisioning for browser-based SSO to facilitate cross-domain account management, one designed for an IdP, and one for an SP:

- At an IdP site, an administrator automatically provisions and maintains user accounts for partner SPs who have implement the System for Cross-domain Identity Management (SCIM) or, when using optional plugin software as a service (SaaS) connectors, for selected hosted-software providers..
- At an SP site, an administrator provisions accounts within the organization automatically from SCIM-enabled IdPs or usesinformation from SAML assertions received during SSO events.

For more information, see User provisioning.

Enterprise deployment features

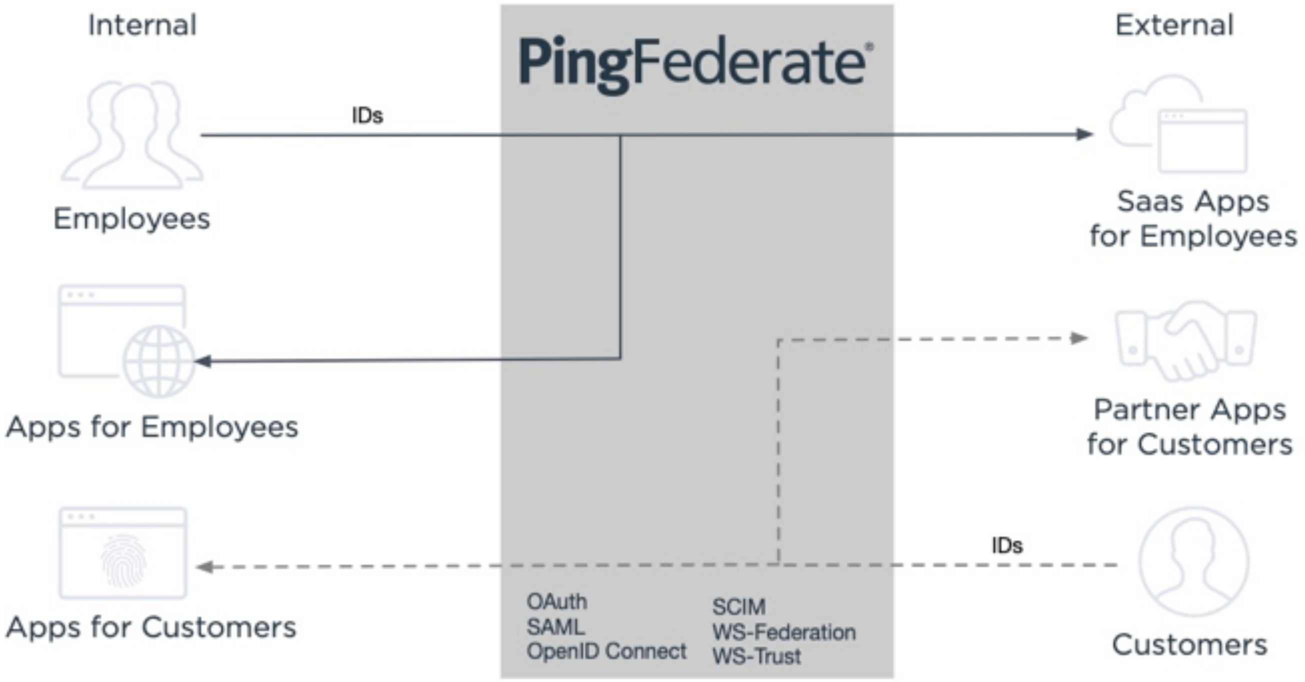
PingFederate’s enterprise-deployment architecture provides one location to manage your environment, eliminating the need to maintain redundant copies of these configurations and trust relationships.

One-time configuration, available everywhere

PingFederate houses all protocol definitions, public key infrastructure (PKI) keys, policies, and profiles in a single location. You can add new protocols, profiles, or use cases, and then make them available to your entire organization.

Improved security

PingFederate improves security by creating a single “doorway” through which all your identity information is exchanged. The same doorway is used regardless of who the users are or in which direction information is shared, such as internal users accessing external applications and external users accessing internal systems.



Load balancers and reverse proxies

To enhance performance and security, you can deploy PingFederate behind an application load balancer (ALB), a network load balancer (NLB), a reverse proxy, or a similar network traffic management solution on-premise or in the cloud.

> **Important**
>
> PingFederate can benefit from knowing the originating IP addresses of incoming requests. Before you deploy PingFederate in the cloud or behind a load balancer, to avoid unexpected behaviors, work with your network team and configure PingFederate’s **Incoming Proxy Settings** accordingly. For more information, see Configuring incoming proxy settings.

Extensive audit and logging capabilities

With PingFederate’s extensive auditing and logging capabilities, you can complete logging-related compliance and service-level requirements, without having to acquire and consolidate disparate logs from throughout your organization.

Business use case driven configuration

The PingFederate administrative console supports various protocol, while reducing complexity and learning curves. You are guided through configuration steps applicable only to the business use cases you need to support.

Additional features

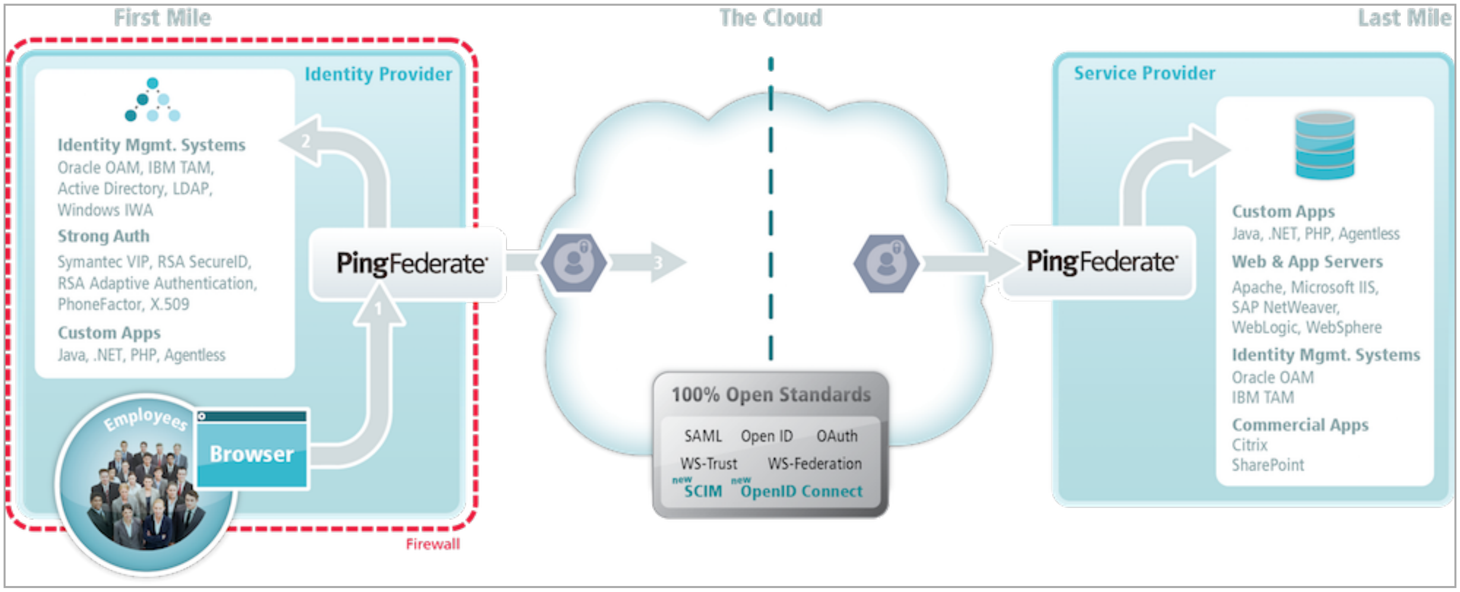
PingFederate's lightweight, standalone architecture allows server integration and partnering with existing home-grown and commercial identity management (IdM) systems and applications providing standards-based single sign-on (SSO) and API security integration benefits.

PingOne

The PingOne Cloud Platform offers one platform for unifying all your digital identities and allowing your users and devices to securely access cloud, mobile, SaaS and on-premises applications and APIs. With PingFederate, you can leverage these services to accelerate the digital transformation of your environment for better security and compliance.

Integration kits

Use PingFederate's software development kit (SDK) for creating custom integrations with existing authentication services and target applications as well as quick connections to various partners. For more information, see Bundled adapters and authenticators. Download PingFederate integration kits from the Ping Identity Downloads website.



Multiple security-domain, multi-protocol federation

Token translators

Ping Identity offers special token processors for an IdP and token generators for a SP that enable the WS-Trust security token service (STS) to validate and issue a variety of token types. These plug-ins supplement built-in SAML token processing and generation and handle the local identity tokens required in many security contexts. For more information, see Token processors and generators.

SaaS connectors

SaaS connectors offer a streamlined approach for browser-based SSO to selected SaaS providers, including automatic user provisioning and deprovisioning. The connector packages include quick-connection templates, which automatically configure endpoints and other connection information for each provider. For more information, see Outbound provisioning for IdPs.

Cloud identity connectors

Offer your users the option to login, register, and access your cloud-based applications with their social identity from sites like Apple, Facebook, Google, LinkedIn, Microsoft, and Twitter.

Key concepts

This section provides background information and preparation to help administrators understand and use PingFederate.

Connection Types

PingFederate features an integrated administrative console for configuring connections to identity-federation partners. The four connection types include:

- Browser-based single sign-on (SSO) – Called Browser SSO in the administrative console, this term refers to standards-based secure SSO, which generally depends on a user's browser to transport identity assertions and other messaging between partner endpoints. For more information, see Supported standards.
- WS-Trust security token service (STS) – Employs the PingFederate STS, which enables web service clients (WSCs) and web service providers (WSPs) to extend SSO to identity-enabled web services at provider sites. For more information, see the WS-Trust STS. These standards, including WS-Trust, do not rely on the user's browser for message transport.
- OAuth Assertion Grant – Exchanges a SAML assertion or a JSON Web Token for an OAuth access token with the PingFederate authorization server (AS). For more information, see About OAuth.
- Provisioning – Provides automated cross-domain inbound and outbound user management. For more information, see User provisioning.

You can configure the types of connections together for the same partner or independently.

WS-Trust STS

PingFederate WS-Trust STS allows organizations to extend SSO identity management (IdM) to web services. For more information see, About WS-Trust STS.

OAuth

You can configure PingFederate to act as an OAuth authorization server (AS), allowing a resource owner to grant authorization to an OAuth client requesting access to resources hosted by a resource server (RS). For more information, see About OAuth.

SSO integration kits and adapters

PingFederate provides bundled and separate integration kits that include adapters that plug into the PingFederate server and agent toolkits that interface with local IdM systems or applications as needed. For more information, see SSO integration kits and adapters.

Security infrastructure

PingFederate security infrastructure supports encrypted messaging, certificates, and digital signing. For more information, see Security infrastructure.

Hierarchical plugin configuration

PingFederate allows you to use a configuration of an adapter, as well as certain other PingFederate plugins, as a parent instance from which you can create child instances. For more information, see Hierarchical plugin configurations.

Identity mapping

PingFederate enables identity mapping between domains for browser-based SSO and WS-Trust STS. For more information, see Identity mapping.

User attributes

Federation transactions require the transmission of a unique piece of information that identifies the user for identity mapping between security domains. For more information, see User attributes.

User provisioning

PingFederate provides cross-domain user provisioning and account management. For more information, see User provisioning.

Customer identity and access management

PingFederate empowers administrators to deliver a secure and easy-to-use customer authentication, registration, and profile management solution. For more information, see Customer identity and access management.

Federation hub use cases

As a federation hub, PingFederate can bridge browser-based SSO between IdPs and SPs. For more information, see Federation hub use cases.

Federation planning

An essential first step in establishing an identity federation involves discussions and agreements between you and your connection partners. For more information, see the Federation planning checklist.

WS-Trust STS

The PingFederate WS-Trust Security Token Service (STS) allows organizations to extend single-sign on (SSO) identity management (IdM) to web services.

You can configure the WS-Trust STS for partner connections independently or in conjunction with browser-based SSO for either an identity provider (IdP) or a service provider (SP) deployment. The STS is bundled with separate plug-ins for standard SAML token processing and generation.

For information about WS-Trust and the role of an STS, see Web services standards.

Connection-based policy

PingFederate employs a partner-connection configuration for both IdP and SP roles, which enables the association of web services authentication policies with federation partners. For more information, see Connection-based policy.

Token processor and generator

PingFederate provides support for a variety of security-token formats through token processors and generators. For more information, see Token processors and generators.

WSC and WSP support

Ping Identity provides the Java client software development kit (SDK) for enabling web service applications to interact with the PingFederate STS. For more information, see WSC and WSP support.

STS OAuth integration

PingFederate STS provides several ways to facilitate the use of issued tokens with an OAuth authorization server (AS). For more information, see STS OAuth integration.

Connection-based policy

For both the identity provider (IdP) and service provider (SP) roles, PingFederate employs a partner-connection configuration, which enables the association of web services authentication policies with federation partners.

For Security Token Service (STS) processing, these policies define configurations for handling WS-Trust requests and transferring identity information between security domains. For more information, see Web services standards.

IdP configuration

Use the administrative console in an IdP role to configure WS-Trust request-processing policy for your SP partner including:

- The type of SAML token to create in response to an issue request from a web service client (WSC) application
- The mapping of attributes to include within the issued SAML token
- The key used to create a digital signature for the issued SAML token

SP configuration

Use the administrative console in an SP role to configure WS-Trust request-processing policy for your IdP partner including:

- Whether to validate the incoming SAML token only, or to validate the incoming token and also issue a local token
- The mapping of attributes to include in the locally issued token when applicable
- The certificate used to verify the digital signature for the incoming SAML token
- The key used to decrypt the incoming SAML token when needed

Token processors and generators

PingFederate provides support for a variety of security-token formats through token processors and generators.

These token processors and generators plug into the PingFederate server and deploy similarly to browser-based single sign-on (SSO) adapters. For more information, see Bundled adapters and authenticators.

For an identity provider (IdP), token processors provide a mechanism through which PingFederate can validate an incoming token from a web service client (WSC) and map attributes to be included in the issued SAML token.

For a service provider (SP), token generators provide a mechanism through which PingFederate can generate a local token based upon the incoming SAML token from a WSP and map attributes to be included in that token.

PingFederate only generates SAML 1.1 or 2.0 token when it is configured as an IdP for sending across trust boundaries to a federate SP partner. PingFederate only accepts SAML tokens when configured as an SP. Token plug-ins allow a modular approach for validating and producing the token types used by different applications or systems within a conceptual trust domain. PingFederate provides bundled and separately available token plug-ins.

> **Tip**
>
> For direct security token service (STS) token exchange within the same domain or trust boundary, use the PingFederate STS to exchange one token type for another directly, without generating a transitional SAML token. For more information, see Token translator mappings.

PingFederate allows you to use a configuration of a token processor or generator as a parent instance from which you create child instances. For more information, see Hierarchical plugin configurations.

Bundled token plug-ins

PingFederate comes installed with token processors for an IdP configuration that accept and validate SAML 1.1 or 2.0 tokens, OAuth bearer access tokens, JSON web tokens (JWT), username tokens, and Kerberos tokens. For more information, see Token models and management. SAML tokens are issued on the IdP side through built-in browser-based SSO capabilities.

For an SP configuration, PingFederate provides token generators for issuing local SAML 1.1 or 2.0 tokens. PingFedearate validates incoming SAML tokens using built-in capabilities.

Commercial token plug-ins

Ping Identity provides token plug-ins called token translators to work with various authentication systems and identity management (IdM) systems. You can download the available plug-ins from the Ping Identity Downloads website.

WSC and WSP support

Ping Identity provides the Java client software development kit (SDK) for enabling web service applications to interact with the PingFederate security token service (STS).

For web service client (WSC) STS clients, PingFederate provides built-in protocol support for Windows Identity Foundation (WIF) applications based on the Windows Communication Foundation (WCF) framework.

> **Note**
>
> The WIF framework includes WS-* protocol support and can interact natively with PingFederate.

Client SDK

The STS Java client SDK provides interfaces that create the WS-Trust Request Security Token (RST) and Request Security Token Response (RSTR) messaging to interact with the PingFederate STS endpoints. Using the SDK library, applications are not responsible for forming these WS-Trust protocol messages, and instead interact only with the tokens themselves.

The SDK is available for download on the Ping Identity Downloads website.

Windows Identity Foundation clients

PingFederate natively supports STS clients using claims-based WIF technology. Claims-based federated identity for web services is a part of the WS-Trust standard that permits client applications to make access-policy decisions, when specifically categorized user attributes are sent in the security token. For more information, see Attribute contracts.

The PingFederate STS supports the following bindings in the .NET federated-security scenarios with WS-Trust:

- `WSFederationHttpBinding`
- `WS2007FederationHttpBinding`

Additionally, the PingFederate STS supports the following bindings for RST and RSTR interactions with .NET. Support for these bindings is limited to the Username, x509, SAML 1.1, and SAML 2.0 token types:

- `WSHttpBinding`
- `WS2007HttpBinding`

> **Note**
>
> For token types such as Kerberos, where customizing default bindings might be necessary, the PingFederate STS supports the use of customBinding.
> For more information about bindings, see Microsoft's System-Provided Bindings.

To expedite configuring their applications, PingFederate provides metadata for developers. PingFederate offers two varieties of metadata, which work together to arrive at functional WSC and web service provider (WSP) configurations:

- STS Metadata Exchange, which contains connection details relating to the SP partner. See the **/pf/sts_mex.ping** section in System-services endpoints.
- Federation Metadata, which contains details on the PingFederate public signing certificate and other information required to establish the trust relationship. See the **/pf/federation_metadata.ping** section in System-services endpoints.

For more information about claim-based federated identity, see Microsoft's A Guide to Claims–based Identity and Access Control.

STS OAuth integration

PingFederate security token service (STS) provides several ways to facilitate the use of issued tokens with an OAuth authorization server (AS).

OAuth token processor

This token processor provides a mechanism through which PingFederate STS can validate an incoming OAuth Bearer access token. The token processor reads and validates the access token and returns any additional user attributes defined.

JWT bearer token grant type

`urn:ietf:params:oauth:grant-type:jwt-bearer`

This token request returns a JSON Web Token (JWT) that a web service client (WSC) can use to request OAuth access tokens from any OAuth AS that supports using JWTs as authorization grants, as defined in JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants specification.

OAuth access token with JWT bearer token grant type

`oauth-v2:access:token:response\|via\|urn:ietf:params:oauth:grant-type:jwt-bearer`

This proprietary token request is similar to the JWT Bearer Token grant type but returns an OAuth access token directly. Acting as an identity provider (IdP), PingFederate generates the intermediate JWT and requests an access token from the OAuth AS on behalf of the WSC. The AS endpoint is obtained from the `AppliesTo` element of the WS-Trust request security token (RST) message.

SAML 2.0 bearer assertion grant type

`urn:ietf:params:oauth:grant-type:saml2-bearer`

This token request returns an encoded SAML assertion that a WSC can use to request OAuth access tokens from any OAuth AS that supports the SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants specification.

OAuth access token with SAML 2.0 bearer assertion grant type

`oauth-v2:access:token:response\|via\|urn:ietf:params:oauth:grant-type:saml2-bearer`

This proprietary token request is similar to the SAML 2.0 Bearer Assertion grant type but returns an OAuth access token directly. Acting as an IdP, PingFederate generates the intermediate, encoded SAML assertion and requests an access token from the OAuth AS on behalf of the WSC. The AS endpoint is obtained from the `AppliesTo` element of the WS-Trust RST message.

These capabilities bridge the WS-Trust client-STS relationship and the trust relationship the same client with an OAuth AS, allowing the client to obtain additional resources on behalf of already-authenticated users in follow-on transactions.

About OAuth

Configuring PingFederate as an OAuth authorization server (AS) allows a resource owner (RO), typically an end user, to grant authorization to an OAuth client requesting access to the resource server (RS).

The OAuth AS issues tokens to clients on behalf of an RO for use in authenticating a subsequent API call to the RS, typically, but not exclusively, a REST API call.

> **Tip**
>
> If your PingFederate license does not include the OAuth AS capabilities, contact sales@pingidentity.com.

You can configure the PingFederate OAuth AS independently or in conjunction with security token service (STS) or browser-based single sign-on (SSO) for either an identity provider (IdP) or a service provider (SP) deployment.

In an IdP deployment, an IdP adapter is used to authenticate and provide user information for the access token. In an SP deployment, the inbound SAML assertion is used to provide authentication information about the user associated with the access token through an OAuth attribute mapping in the IdP connection.

For an STS IdP, PingFederate provides an OAuth token processor that validates incoming OAuth Bearer access tokens.

Delegated access types

To enhance access control, PingFederate supports both explicit and implicit delegation of transaction approval.

Explicit delegation

This is the most common OAuth use case, which involves a resource owner (RO) who explicitly delegates the authority to a client to make API calls to a resource server (RS) and is asked to approve the transaction. This is the type of delegation inherent in web redirect flow.

Implicit delegation

Implicit delegation also generally involves a client who calls an API on behalf of a user. However, the client's authority is implied by the nature of the transaction, and the user is not specifically asked to approve the transaction.

Token models and management

Successful OAuth transactions require an OAuth authorization server (AS) to issue tokens characterized by both security model and data model for use in authenticating an API call.

Token security model

A token security model refers to the conditions that must be met by a client to use a token on an API call. The currently supported model is a Bearer Token. A client's presentation of the token – for example, as a parameter on the API call – to the resource server (RS) is interpreted as providing sufficient proof to the RS that the client received the same token from the OAuth AS.

Token data model

A token data model refers to whether the token carries identity and security information or acts as a pointer to the information.

Self-contained tokens (JSON Web Tokens)

Contain identity and security information and attributes in a transport format such as JSON, signed by the AS and verified directly by the RS.

Reference tokens (Internally Managed Reference Tokens)

Serve as a reference to some set of attributes. The RS must de-reference the token for the corresponding identity and security information at the OAuth AS that issued it.

Token management

PingFederate supports multiple access token management instances, providing flexibility for enterprises where deployments require different token data models, token lifetimes, attribute contracts, token validation rules, or any combination of them for various clients.

Related links

- Access token management

Grant types

To obtain an access token, a client interacts with an OAuth authorization server (AS), sending a request for an access token that includes an access grant. An access grant is also used when a resource server (RS) requests validation of an access token from the AS.

Primary grant types

OAuth defines several different access grant types. Each grant type reflects different authorization mechanisms.

Authorization code

`authorization_code`

An authorization code is returned to the client through a browser redirect after the resource owner (RO) gives consent to the AS. The client subsequently exchanges the authorization code for an access token and often a refresh token. RO credentials are never exposed to the client.

Resource owner password credentials

`password`

The client collects the RO’s password and exchanges it at the AS for an access token and often a refresh token. This grant type is suitable in cases where the RO has a trust relationship with the client, such as its computer operation system or a highly-privileged application because the client must discard the password after using it to obtain the access token.

Refresh token

`refresh_token`

A refresh token often returns with an access token. Once the original access token expires, the corresponding refresh token is sent to the AS to obtain a fresh access token and fresh refresh token without requiring the RO to re-authenticate. This allows short-lived access tokens to exist between the client and the resource server and long-lived tokens between the client and the AS.The refresh token grant type only works in conjunction with either the authorization code or RO password credentials grant type.

> **Note**
>
> When an attempt to replay an invalid refresh token is noticed, PingFederate revokes the entire chain of refresh and access tokens, forcing a re-authentication, subject to sessions and persistent grant settings.

Implicit

`implicit`

A browser redirect responds to the RO authorization request, rather than an intermediate authorization code, and returns an access token to the client. This grant type works for clients incapable of keeping client credentials confidential for use in authenticating with the AS, such as client applications implemented in a browser using a scripting language like JavaScript.

Client credentials

`client_credentials`

The client presents its own credentials to the AS to obtain an access token. This access token is either associated with the client's own resources, and not a particular RO, or with a RO for whom the client is otherwise authorized to act. This is the grant type to use for machine-to-machine authentication, server-to-server authentication, and worker applications.

Extension grant types

OAuth provides an extension mechanism for defining new extension grant types to support additional clients or to provide a bridge between OAuth and other trust frameworks. An OAuth client uses an extension grant type by specifying an absolute URI as the value of the `grant_type` parameter and by adding any additional parameters necessary when contacting the token endpoint at `/as/token.oauth2`.

PingFederate supports the following extension grant types:

Assertion grants

JWT Bearer

`urn:ietf:params:oauth:grant-type:jwt-bearer`

The client obtains a JSON web token (JWT) and uses it to request an access token from the AS. This grant type allows a client to use an existing trust relationship, expressed through a JWT, without a direct user approval step at the AS.

SAML 2.0 Bearer

`urn:ietf:params:oauth:grant-type:saml2-bearer`

The client obtains a SAML 2.0 Bearer assertion and uses it to request an access token from the AS. Similar to the JWT Bearer grant type, this grant type allows a client to use an existing trust relationship, expressed through a SAML assertion, without a direct user approval step at the AS.

> **Note**
>
> The SAML assertion used for this grant type generally cannot be a browser-based SSO assertion. To ensure its validity, the assertion must be associated with WS-Trust security token service (STS) processing.

Client-initiated backchannel authentication (CIBA) grant

`urn:openid:params:grant-type:ciba`

The client presents an identity hint to the AS. The AS identifies the RO based on the hint provided and then authenticates and obtains authorization from the RO through an out-of-band flow. Depending on the setup, the client either polls the AS for the authorization result or wait for a signal from the AS to return to the AS for the authorization result. If the RO approves the authentication request, the AS returns an access token to the client. Otherwise, the AS returns an error message per the specification.

Device authorization grant

`urn:ietf:params:oauth:grant-type:device_code`

The client presents a device code and user code to the AS to identify the device-authorization session and obtain an access token. This access token is associated with a RO for whom the client is otherwise authorized to act.

token exchange grant

`urn:ietf:params:oauth:grant-type:token-exchange`

The client presents a security token to the AS. In exchange, the AS returns another kind of security token to the client. The new token might be an access token that is more narrowly scoped for a downstream service or it could be an entirely different kind of token. This grant type supports subject and actor tokens employing impersonation and delegation.

Validation grant

`urn:pingidentity.com:oauth2:grant_type:validate_bearer`

This proprietary PingFederate OAuth extension enables an RS to act as a client in the request/response exchange with PingFederate as the AS in this scenario. The grant type allows an RS to check with PingFederate on the validity of a bearer access token received from a client making a protected-resources call.

Scopes

In addition to OAuth, PIngFederate supports the use of scopes to constrain and define access privileges.

OAuth provides a mechanism to constrain the privileges associated with an access token, whereas scopes provide a way to more specifically define the privileges requested and granted. Generally, a client specifies the desired scopes when sending an authorization request to the authorization server. If the user approves, the authorization server issues an access token with these scopes.

Scopes are configured globally using the **System > OAuth Settings > Scope Management** configuration wizard. Once defined, you can manage the availability of scopes on a client-by-client basis.

Static scopes and dynamic scopes

As an authorization server, PingFederate supports the concepts of static scopes and dynamic scopes. To define a static scope, use a text value such as `read_bank_account`. To define a dynamic scope, use a text value with a variable component represented by a wildcard, such as `read_bank_account_txn:*`. As illustrated, dynamic scopes allow clients to request authorization using scope values with a variable component from one request to another.

Related links

- Scopes and scope management

Consent approval

With Authorization Code, Implicit, and Device Authorization grant types, an authorization server (AS) prompts the user to grant authorization to share user information. Once granted, the AS issues an access token to the client who uses it to access information from the resource server (RS).

Default consent user interface

PingFederate handles the consent approval process by presenting the **Request for Approval** window to the user by default. This window displays a list of requested permissions, `scopes`, along with their descriptions as configured in PingFederate. It is up to the user to approve or deny individual scopes.

External consent user interface

As use cases evolve towards giving users more control over their data, it becomes more important to provide detailed information about the requests. While the scope description can help, PingFederate also supports the use of an external web application to prompt for authorization consent. This approach opens up the opportunity to retrieve additional information specific to the users. For example, the web application can be written in such a way that when a client requests the `read_bank_account` scope, the web application retrieves the user's customer information file and gives the user the ability to choose which accounts to make available to the client.

Related links

- Grant types
- Configuring authorization server settings
- External consent user interface

Client management and storage

OAuth clients interact with an authorization server (AS) to obtain access tokens and optionally refresh tokens to access protected resources on resource servers.

PingFederate provides administrators the flexibility to manage OAuth clients using the following interfaces:

- The administrative console
- The administrative API
- The OAuth Client Management Service

Additionally, PingFederate supports dynamic client registration based on the OAuth 2.0 Dynamic Client Registration Protocol specification.

Storing client records in XML files by default allows administrators to manage clients using the administrative console and the administrative API. It also allows developers to submit client creation requests based on the Dynamic Client Registration protocol specification. The configuration archive contains client records.

Alternatively, because the OAuth Client Managment Service requires external storage of client records, PingFederate supports configuration to store client records externally on a database server, a directory server, or some other storage medium through the use of the PingFederate SDK. Under this configuration, the configuration archive does not include client records.

Related links

- Managing OAuth clients
- PingFederate administrative API
- OAuth Client Management Service
- Dynamic client registration
- OAuth client datastores
- Configuration archive

Client authentication schemes

Most OAuth and OpenID Connect use cases require the client application to authenticate successfully before its requests can be processed further.

As an OAuth authorization server (AS), PingFederate supports the following client authentication schemes:

- Client secret for HTTP Basic authentication
- Client TLS certificate for mutual TLS authentication
- Private key JWT for the private_key_jwt client authentication method, as defined in the OpenID Connect specification
- Client secret JWT for the client_secret_jwt client authentication method, as defined in the OpenID Connect specification
- None when authentication is not required

When deployed as an OpenID Connect Relying Party (RP), PingFederate authenticates through client secret and private key JSON web tokens (JWT). It also handles the scenario where authentication is not required.

Related links

- Managing OAuth clients
- OpenID Connect Relying Party support

Dynamic client registration

As an OAuth provider, PingFederate supports a number of OAuth protocols.

PingFederate supports dynamic client registration based on the OAuth 2.0 Dynamic Client Registration Protocol specification. When enabled, it allows developers to register OAuth clients through an API based on open standards.

Related links

- Configuring client settings

Transient grants and persistent grants

There are two types of OAuth authorization grants: transient grants and persistent grants.

Transient grants

Transient grants are valid only for the lifetime of their respective access tokens. Transient authorizations include those obtained by OAuth clients in the following manners:

- Grants obtained by using the authorization code, resource owner credentials, or device authorization grant type, without the refresh token grant type
- Grants obtained by using client credential, JWT bearer, SAML 2.0 bearer assertion, or token exchange grant type

Transient grants are not preserved.

Persistent grants

Persistent grants typically bear a longer lifetime than their respective access tokens do. Persistent authorizations include those obtained by OAuth clients in the following ways:

- Grants obtained or updated using the authorization code, resource owner credentials, or device authorization grant type, in conjunction with the refresh token grant type

  > **Note**
  >
  > If the use cases involve mapping attributes from authentication sources, such as IdP adapter instances or IdP connections, or password credential validator (PCV) instances to the access tokens, directly or through persistent grant-extended attributes, storing these attributes from authentication sources and their values along with the persistent grants maintains them for reuse when clients subsequently present refresh tokens for new access tokens.

- Grants obtained or updated by using the implicit grant type, for which PingFederate is configured to reuse existing persistent grants

  > **Note**
  >
  > If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens, runtime procedures obtain attribute values for each token request, but persistent grants do not store with attributes or their values.

Persistent grant lifetime and maintenance

Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

Grants persist without any expiration information. Grants also persist with an idle timeout window, a maximum lifetime, or both. If you configure an idle timeout value, the idle timeout window slides when a persistent grant updates. When you have an idle timeout value configured without a maximum lifetime, persistent grants remain valid until they expire because of inactivity or until the grant storage revokes or removes them. When you have an idle timeout value configured with a maximum lifetime, persistent grants remain valid until they expire due to inactivity or lifetime expiration or until the grant storage removes them.

> **Note**
>
> Grants can persist without expiration information, or with expiration depending on your PingFederate configuration. You should set a maximum lifetime for a grant.
> Grants can persist with an idle timeout window, a maximum lifetime, or both. If you configure an idle timeout value, the idle timeout window restarts when a persistent grant updates.
> When a persistent grant has an idle timeout value, but no maximum lifetime, the grant remains valid and will never expire or be purged, even if there is no activity. These grants must be explicitly revoked.
> When a persistent grant has both an idle timeout and maximum lifetime configured, the grant remains valid until it expires due to inactivity, or until its lifetime expires. Inactive grants will be purged by the grant storage. It can also be explicitly revoked.

PingFederate removes expired grants and the associated attributes from the grant datastore once a day. The frequency and the size of the cleanup batch are configurable. Depending on the datastore being used, the removal of expired grants can be done by the datastore itself. You should configure the datastore to purge expired grants whenever possible. Optionally, PingFederate caps the number of persistent grants on a basis of the combination of user, client, and grant type.

Persistent grant storage

Support for persistent grants requires the use of a database server or a directory server for long-term storage. PingFederate also supports other storage solutions through the PingFederate SDK. For more information, see OAuth grant datastores.

PingFederate uses a built-in HSQLDB database as its grant datastore after the initial setup.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Related links

- Grant types

Grant storage and management

PingFederate uses a built-in HSQLDB database as its persistent grant datastore after the initial setup.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

PingFederate removes expired grants and the associated attributes from the grant datastore once a day. The frequency and the size of the cleanup batch are configurable. Depending on the datastore being used, the removal of expired grants can be done by the datastore itself. You should configure the datastore to purge expired grants whenever possible. Optionally, PingFederate caps the number of persistent grants on a basis of the combination of user, client, and grant type.

For revocation, PingFederate provides two endpoints.

Token revocation endpoint

The token revocation endpoint allows clients to notify the authorization server that they no longer need a previously-obtained refresh or access token. The revocation request invalidates the actual token and possibly other tokens based on the same authorization grant.

Grant-management endpoint

The grant-management endpoint allows resource owners to view and optionally revoke the persistent access grants they have authorized.

Intended for OAuth clients, the token revocation endpoint is the endpoint to which clients send their token revocation requests. The grant-management endpoint is for resource owners. It displays a list of grants the resource owners have made. Resource owners can view and optionally revoke one or more grants as they see fit.

Related links

- OAuth grant datastores
- OAuth persistent grants cleanup
- Token revocation endpoint
- Grant-management endpoint

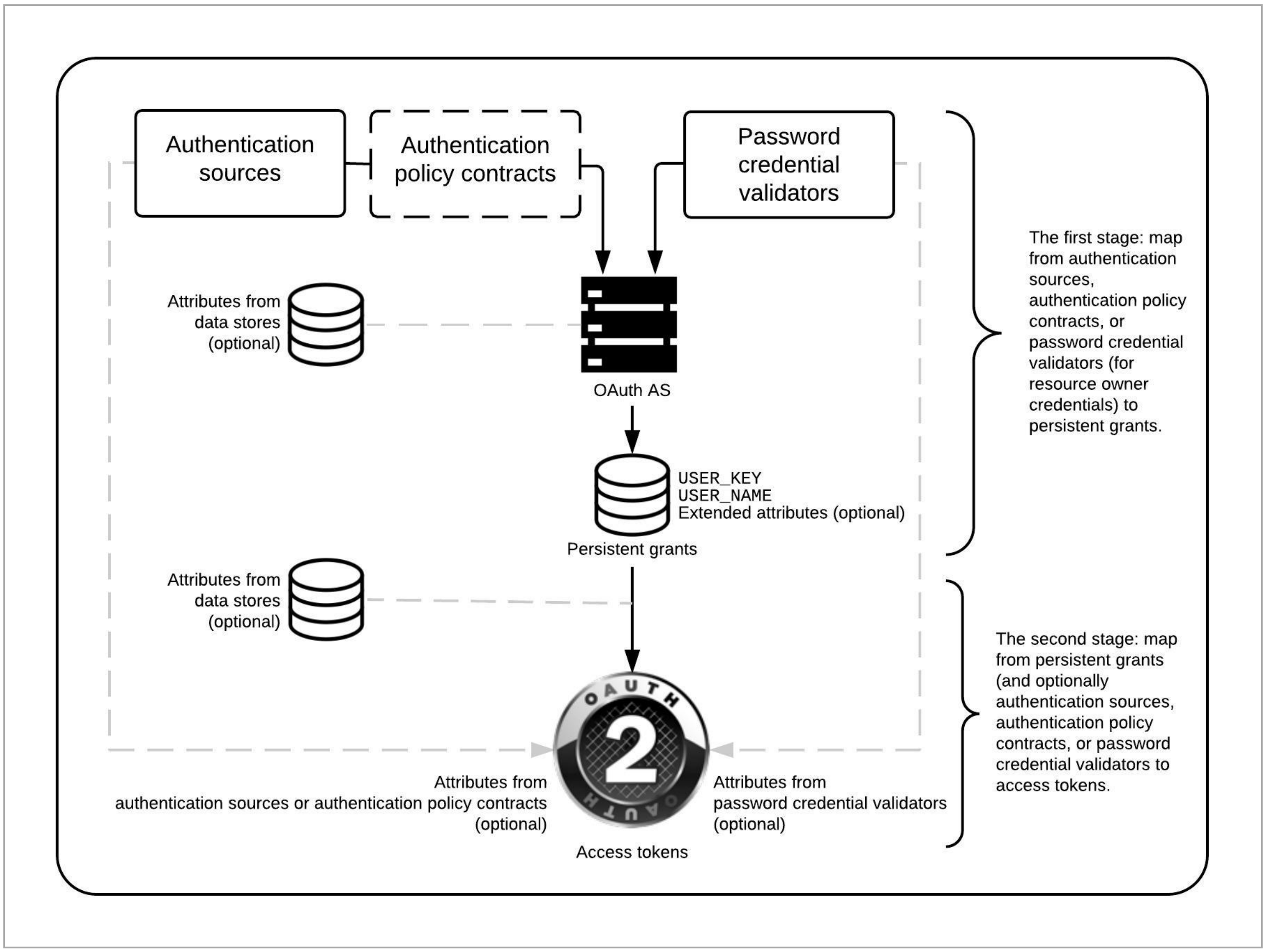
Mapping OAuth attributes

Mapping OAuth attributes is a two-stage processing workflow.

The two stages of mapping OAuth attributes include:

- The first stage: map from authentication sources, such as IdP adapter instances or IdP connections, authentication policy contracts, or Password Credential Validator instances (for resource owner credentials) to persistent grants.
- The second stage: map from persistent grants, authentication sources, authentication policy contracts, authentication context, or Password Credential Validator instances to access tokens.

> **Note**
>
> This two-stage mapping workflow is different from other mapping scenarios in PingFederate, which involve just a one-phase configuration.


Authentication sources
Authentication policy contracts
Password credential validators
Attributes from data stores (optional)
OAuth AS
USER_KEY
USER_NAME
Extended attributes (optional)
Persistent grants
Attributes from data stores (optional)
Attributes from authentication sources or authentication policy contracts (optional)
Attributes from password credential validators (optional)
Access tokens
The first stage: map from authentication sources, authentication policy contracts, or password credential validators (for resource owner credentials) to persistent grants.
The second stage: map from persistent grants (and optionally authentication sources, authentication policy contracts, or password credential validators to access tokens.

The first stage

To accomplish the first stage, setting up persistent grants requires mapping, including a user key and all extended attributes.

The mappings use attributes obtained during initial authentication events within PingFederate, namely attributes from IdP adapter instances, attributes from assertions through IdP connections, attributes from authentication policy contracts, or attributes returned by password credential validator instances. Configure data store queries to accomplish different tasks, such as retrieving the user identifier from an LDAP directory server as the user key.

> **Important**
>
> The `USER_KEY` attribute values must be unique across all end users, because the `USER_KEY` attribute is the user identifier to store and to retrieve persistent grants. For example, the `sAMAccountName` attribute value of an end user in one domain might match that of another end user in another domain. In this case, you can map the `Subject DN` attribute to the `USER_KEY` attribute.

The second stage

The second mapping configuration involves mapping from persistent grants the user keys, any extended attributes derived from the first stage, or both into OAuth access tokens.

When the authentication context matches specific mappings between the authentication sources, authentication policy contracts, or password credential validators, the attributes map into access tokens.

The **HTTP Request** Java object retrieves the authentication method that a client uses, or the private key JWT for client authentication if the client uses the private_key_jwt authentication method.

The HTTP Request Java object maps it into the access tokens.

Data stores used here retrieve any required user attributes.

Runtime processing

At runtime, the first time a client requests an OAuth token, the process employs the two mapping sequences. Every time an existing persistent grant requests a new access token, the process invokes the second mapping.

Related links

- Grant contract mapping
- Token mapping

OAuth user-facing windows

The PingFederate OAuth authorization server (AS) presents five windows to end users during OAuth transactions.

The windows include:

- One prompting for approval of the requested scopes
- One providing a means of revoking persistent access grants
- Three others displaying information for the purpose of connecting OAuth-capable internet of things (IOT) based on the OAuth 2.0 Device Authorization Grant specification

Administrators can customize these windows as needed.

Related links

- Customizable user-facing pages

OpenID Connect

As an extension of OAuth capabilities, PingFederate supports an optional configuration for OpenID Connect, a modern protocol for secure, lightweight transfer of authentication and user attributes.

OpenID Provider support

As an OpenID Provider (OP), PingFederate supports both the Basic Client and Implicit Client profiles defined in the standard. In both profiles, the end result releases an ID token and an OAuth access token; however, depending on associated grant types, PingFederate might also release a refresh token.

The ID token is an integrity-secured, self-contained token in JSON Web Token (JWT) format containing claims about the user. A client uses the ID token to identify the user accessing the client application through an OP. A client may subsequently use the OAuth access token to retrieve additional claims about the user, such as a complete profile containing full name, email, phone, and other schema elements defined in an OpenID Connect policy from the Userinfo endpoint (`/idp/userinfo.openid`).

For session management, PingFederate provides a front-channel endpoint for OAuth clients using the OpenID Connect protocol to close other associated sessions at `/idp/startSLO.ping` and a back-channel web service for clients to revoke end-user sessions at `/pfws/rest/sessionMgmt/revokedSris` .As an OP, PingFederate optionally accepts request parameters through self-contained, signed JWTs. This capability enables PingFederate to validate the integrity of the request parameters it receives before processing the request further. PingFederate also includes a state hash (`s_hash`) in the ID token to protect the integrity of the `state` parameter.

Relying Party support

As a Relying Party (RP), PingFederate is capable of leveraging identities from OPs to complete browser-based SSO requests. In this use case, PingFederate is the requesting OAuth client application.

The setup involves establishing an IdP connection to the OP. PingFederate retrieves identity information from the OP and passes the end-user claims, which are user attributes in an ID token, to one or more target applications. This configuration allows administrators to take advantage of their existing last-mile integration and expand the horizon of their applications to additional partners using the OpenID Connect protocol.

PingFederate is also capable of sending request parameters through self-contained, signed JWTs, thus adding a layer of security to the transmission of the request parameters. Additionally, if the ID token contains a state hash, PingFederate validates it.

Related links

- What is OpenId Connect?
- OpenID Connect specification, Passing a Request Object by Value
- Financial Services – Financial API - Part 2: Read and Write API Security Profile

CORS support for OAuth endpoints

PingFederate supports cross-origin resource sharing (CORS) for several OAuth endpoints.

The supported OAuth endpoints include:

- `/as/token.oauth2`
- `/as/revoke_token.oauth2`
- `/idp/userinfo.openid`
- `/pf-ws/rest/oauth/grants/`
- `/pf/JWKS`
- `/.well-known/openid-configuration`
- `/as/bc-auth.ciba`

As needed, administrators can add or remove allowed origins using the administrative console on the **Authentication Application** page. For instructions on how to add and remove allowed origins, see Configuring authentication applications.

Once configured, client-side web applications from the trusted origins are allowed to make requests to the PingFederate authorization server for the purpose of accessing protected resources, such as obtaining or renewing access tokens with refresh tokens, presenting access tokens for revocation, querying additional claims (user attributes), and retrieving OpenID Provider configuration information and JavaScript Object Notation (JSON) Web Key Sets.

Related links

- Configuring authorization server settings
- W3C's recommendation on Cross-Origin Resource Sharing (www.w3.org/TR/cors)

Security infrastructure

This section describes the PingFederate security infrastructure that supports encrypted messaging, certificates, and digital signing.

PingFederate's configuration windows integrate these functions to provide complete control over certificate generation and authentication verification.

Digital signatures

A digital signature is a way to verify the identity of a person or entity who originates an electronic document and ensure that the message has not been altered.

Both SAML, including security token service (STS) tokens, and WS-Federation electronic documents use digital signatures.

Handling a digital signature involves message signing, signature and certificate validation, and signing-policy coordination between connection partners.

Message signing

Certificates contain information about the certificate owner along with a public key. Applying a digital signature creates and encrypts a hash from the signing message using the private key.

PingFederate provides a choice of signature encryption algorithms when you require a stronger algorithm.

To ensure the integrity of SAML messages or security token service (STS) tokens, we recommend digital signing practices using public/private keypairs in conjunction with X.509 certificates.

> **Note**
>
> Digital signatures do not encrypt the contents of a message; instead, messages use XML encryption when needed.

Ping Identity recommends a certificate signed by a certificate authority (CA); however, PingFederate will work with self-signed or untrusted third-party-signed certificates. After generating a keypair and a self-signed certificate, use PingFederate to create a certificate signing request (CSR) and send it to a CA for signing. After the CA has generated a CSR, import it into PingFederate's certificate management system. PingFederate's trusted store or the Java runtime `cacerts` store must contain the CA's certificate.

PingFederate enables signing and validation of requests and responses. Additionally, PingFederate provides for certificate generation, import and export functionality, CSR generation, and application of digital signatures. You have the option to create reusable global signing certificates across your federated connection base and import signature verification certificates for each partner. For more information, see Manage digital signing certificates and decryption keys.

> **Note**
>
> Ping Identity recommends generating unique certificates for each connection, which limits exposure if the private key becomes compromised.

Signature validation

After receiving a signed message, PingFederate verifies the signature using the public key that corresponds with the private key used to sign the message or token. Verification involves creating a hash of the received message, using the signing partner's public key to decrypt the hash sent with the original message, and verifying that both hash values are equal.

Certificate validation

PingFederate always checks certificates to see if they have expired when they are initially imported. It also checks certificates at runtime when they are used to verify incoming signed assertions.

PingFederate also checks to see whether a certificate has been revoked, using either certificate revocation lists (CRLs) or the online certificate status protocol (OSCP). Depending on the content of the certificate in question and your requirements, the server will perform either of these checks during single sign-on (SSO) or single log-out (SLO) processing for the following cases:

- Signature verification
- Validation of a client certificate used for authentication to PingFederate when the server is handling direct client requests
- Validation of the server SSL certificate when PingFederate acts as the client making an HTTPS request to a separate server

If the system encounters an expired or revoked certificate, the associated SSO or SLO transaction fails at runtime and writes an error to the transaction log. In the administrative console, the Status column of the respective Certificate Management list identifies the expired or revoked certificate.

In PingFederate, by default, OCSP revocation checking is enabled and CRL revocation checking is disabled. However, you can configure PingFederate to use only CRL checking or as a backup to OCSP checking.

For more information, see Configuring certificate revocation.

CRL revocation checking

CRL revocation checking involves querying a CRL distribution-point URL and ensuring that a certificate is not on the returned revocation list maintained at the site. The certificate specifies the URL.

OCSP revocation checking

OCSP revocation checking provides a more centralized and potentially more reliable means of checking certificate status than CRL revocation checking provides. In this scenario, the incoming certificate embeds an OCSP Responder URL or a configured default URL to query the certificate status.

The primary difference between OCSP and CRL checking is how the verification occurs. CRL checking requires the requesting client to determine if the certificate has been revoked, or if any of the certificates in the chain of issuer certificates has been revoked, based on the returned CRL. With OCSP, the client sends the certificate itself, and the Responder server handles revocation checking to return the certificate status.

For more information about OCSP, see tools.ietf.org/html/rfc2560.

Digital signing policy coordination

To coordinate digital signature policy, partners must first agree about whether they will sign SAML messages or tokens.

In some cases, such as SAML security token service (STS) tokens and single sign-on (SSO) assertions sent across the POST binding, the protocol specifications require signatures. The PingFederate administrative console and the runtime protocol engine enforce these requirements. Some partner connections specify other, optional uses of the digital signatures between partners.

If a trusted, self-signed certificate authority (CA) does not issue a digital-signing certificate, then the signing partner must send the public-key certificate out-of-band, such as through email, to the partner. The partner must import the certificate into PingFederate when configuring a connection to the signing partner for SSO, single log-out (SLO), or STS.

If a trusted CA signs the certificate and the signing partner chooses to embed the certificate in all signed messages, then the verifying partner uses the embedded certificate for signature verification, after validating it against the Subject Distinguished Name (DN) of the original certificate. Because validation only requires the Subject DN, the public-key certificate might not be sent out-of-band.

> **Tip**
>
> During the signature verification configuration, PingFederate extracts the Subject DN from the certificate when available.

The next section provides more information about the two alternative signature-verification trust models described above from the standpoint of the verifying partner.

Trust models

For validating digital signatures, PingFederate provides a selection of trust models in the administrative console for each partner connection, based on the certificate categories listed below. For each trust model, PingFederate always verifies the currency of the certificate and the validity of the message in the certificate specified. Additional checks depend upon the trust model selected.

Anchored certificate

In this case, certificates used for signature verification must be issued by a trusted CA, and the certificate chain must be verifiable recursively back to the root issuer. PingFederate validates the certificate, including recursive revocation checking (when enabled) back to the issuer,for all signed messages from the partner. By default, PingFederate also prompts for the Issuer DN of the certificates to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.

In addition, when choosing the anchored trust model, the incoming message must include the verification certificate for the signature. PingFederate uses that certificate to verify signatures from the partner if its Subject DN matches the partner's public certificate (as specified in the administrative console), the Issuer DN (if specified) matches one of the issuers in the chain, and the Issuer CA certificate is part of the trusted store. This feature provides a dynamic trust model that overcomes the problem of interrupting service to change out expired certificates.

Unanchored Certificate

In the unanchored certificate model, incoming signature verification exclusively uses the certificates imported for a connection into PingFederate or a secondary, backup certificate, either self-signed or trusted CA-issued, when specified. This verification does not apply to the certificate chain. However, when enabled, existing chains receive revocation checking as far as available.

Secure sockets layer

A certificate authority (CA)-signed SSL certificate identifies one or both ends of the federation. SSL/TLS provides an encrypted connection between the two parties to avoid exposing the content of a message. This promotes confidentiality and message integrity.

SAML SSL and TLS scenarios

SSL/TLS should be used in association with the SOAP responder URL and Single Sign-on Service located at an identity provider (IdP) site. On the service provider (SP) side, the Artifact Resolution Service should also use SSL/TLS. Optionally, SSL/TLS can also be used to secure communication between internal data stores and PingFederate and between the PingFederate security token service (STS) and web service client or provider applications.

The SSL/TLS server-client handshake involves negotiating cipher suites to use for encryption and decryption on each side of a secured transaction. You can find cipher suites in the following configuration files:

- `com.pingidentity.crypto.SunJCEManager.xml`
- `com.pingidentity.crypto.AWSCloudHSMJCEManager.xml`
- `com.pingidentity.crypto.LunaJCEManager.xml`
- `com.pingidentity.crypto.NcipherJCEManager.xml`
- `com.pingidentity.crypto.BCFIPSJCEManager.xml`

These cipher-suite configuration files are located in the `<pf_install>/server/default/data/config-store` directory. These files comment out weaker cipher suites. To ensure the most secure transactions, retain this cipher-suite configuration.

> **Important**
>
> Because of the import restrictions of some countries, Oracle Server Java SE Runtime Environment (JRE) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, enable the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy. For more information, see the Java 8 release notes in Oracle's documentation.
> For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide in Oracle's documentation.

Starting with version 9.1, PingFederate selects cipher suites based on the order that they appear in the cipher-suite configuration file for new installations. For upgrades, enable the same selection mechanism. For more information, see Managing cipher suites.

Authentication

PingFederate browser-based single sign-on (SSO) uses three methods to authenticate connection partners making SOAP requests. For STS client SOAP authentication, configure a separate option using either or both of the first two methods listed here. Partners must agree upon the selection of methods and synchronize within IdP and SP federation implementations.

HTTP Basic authentication

Partners identify themselves by passing username and password credentials.

SSL client certificate authentication

Partners use SSL client certificates presented during SOAP request transactions. Each partner needs to import the other partner's certificate out-of-band For more information, see Manage SSL client keys and certificates.

Digital signatures

Partners sign the XML message transmitted through the SSL/TLS connection. The receiver verifies the signatures based upon the certificates configured for that connection. Each partner should import the others' certificates out-of-band. For more information, see Manage digital signing certificates and decryption keys.

Trusted certificates

PingFederate validates the trust of all certificates. PingFederate trusts certificates if the issuer's certificate is also in PingFederate's trusted certificate store. You must import the root certificate of the CA into PingFederate's trusted certificate store or into the Java runtime `cacerts` store.

Encryption

PingFederate supports the optional SAML 2.0 specification allowing for encryption of assertions, including security token service (STS) SAML tokens, which further enhances confidentiality when required.

For SAML 2.0 single sign-on (SSO) connections, you can choose to encrypt entire assertions or individual user attributes, including the user's name identifier. You can use signature verification and signing keys to encrypt and decrypt messages, respectively.

Hierarchical plugin configurations

PingFederate allows you to use a configuration of an adapter along with other PingFederate plugins as a parent instance for any created child instances.

After this, you can then modify the inherited configuration for child instances as needed. This feature provides easier management of adapter settings in cases where only small changes to an existing adapter or plugin configuration need to be made for a particular use case.

For example, different service provider (SP)-connection adapter instances might have their own identity provider (IdP) sign-on URLs, for branding or other application ownership reasons, while the majority of the other adapter configuration settings are the same. In this case, you might want to use a parent/child configuration to override the logon URLs.

> **Tip**
>
> Override adapter instances as part of mapping them into either SPor IdP connections for cases where overridden settings apply only to one particular connection configuration.

Any changes to a parent configuration propagate to its child or connection-based configurations so long as the derived instance has not overridden the changes.

In addition to adapters, PingFederate allows you to create parent/child configurations for the following plugin types:

- Token Translators (see Token processors and generators)
- Access Token Management instances (see Access token management)
- Password Credential Validators (see Password Credential Validators)
- Identity Store Provisioners (see Configuring Identity Store Provisioners)

Identity mapping

Identity mapping is at the core of identity federation. One of the primary goals of SAML is to provide a way for an identity provider (IdP) to send a secure token, called the assertion, containing user-identity information that a service provider (SP) translates or maps to local user stores.

For browser-based single sign-on (SSO), PingFederate enables two modes of identity mapping between domains: account linking and account mapping.

For WS-Trust security token service (STS), PingFederate uses account mapping.

See subsequent topics for more information about these identity mapping options.

Related links

- Supported standards

Account linking

Under the standards, use account linking for browser-based single sign-on (SSO) in cases where each domain maintains separate accounts for the same user.

Account linking uses the SAML assertion to create a persistent association between these distinct user accounts. The account link, or name identifier, such as an email address or identity provider (IdP)-generated pseudonym, identifies individual users. When privacy is a concern, use pseudonyms because they prevent tracing back to a user's identity at the partner site.

During the user's first SSO request, the service provider (SP) prompts for local credentials, which enables the SP to link the name identifier contained within the assertion—either an open attribute or a pseudonym—with the user's local account. Subsequent SSO events will not prompt the user to authenticate with the SP because the SP federation server keeps a table associating remote users' name identifiers with local user accounts. The SP associates the link to the user's corresponding local account and provides access to the account without separate authentication.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

The name identifier optionally includes additional attributes. When using a pseudonym as the account link, take care to send only general attributes, such as a user's organizational role or department, that will not compromise privacy.

Linking permission and defederation

The SAML specification also allows the SP application to build in user verification and approval of account linking and provides a means for the user to permanently cancel the linking, known as defederation. For more information, see developers_reference_guide:pf_sp_services.adoc#spDefederatePing. A defederated user might later elect to re-associate with a local user account.

SP affiliations

Under the SAML 2.0 specifications, an IdP configures PingFederate to enable a group of SPs, called an SP affiliation, to share the same persistent name identifier. For more information, see SP affiliations. An SP affiliation facilitates the use case where a number of business partners have an existing relationship and where sharing a single name identifier among all parties reduces the federation integration effort.

Account mapping

Account mapping, also called attribute mapping, enables a service provider (SP) to use PingFederate to perform a user lookup and map a user's identity dynamically based on one or more attributes received in the assertion.

Looking up the user always exposes the attributes. In other words, both the identity provider (IdP) and the SP know these attributes, such as an email address.

Account mapping achieves one-to-one mapping where individual user accounts exist on both sides of a federated connection or many-to-few mapping where IdP users without accounts at destination sites map to guest accounts or to a role-based general account.

For browser-based single sign-on (SSO), transient identifiers provide an additional level of privacy—virtual anonymity—by generating a different opaque ID each time the user initiates SSO. Transient IDs are often used in conjunction with federation role mapping to map the user to a guest account or to a role-based account based on the user's association with the IdP organization rather than personal attributes.

As with pseudonyms, additional attributes might be sent with the transient identifier. Again, take care to preserve privacy.

In B-to-B or B-to-E use cases where an administrator might create a user lookup on behalf of the user, the administrator might implement account mapping.

User attributes

Federation transactions require, at a minimum, the transmission of a unique piece of information, such as an email address, that identifies the user for identity mapping between security domains.

In addition to attributes used for identity mapping, the identity provider (IdP) can pass other user attributes in an assertion, including SAML tokens for web services. The service provider (SP) uses this supplemental information for several purposes. For example, the SP can use attributes to map and authorize the user into a specific role with associated site permissions or to customize the end application display for a more robust user experience.

The SP can also incorporate additional attributes prior to creating a session for the target application. This is common where the SP also maintains an account for the user and wants additional information for profiling or access-policy purposes.

Attributes must be carefully managed between IdPs and SPs. PingFederate facilitates the process by providing configuration steps that enable administrators to:

- Define and enforce attribute_contract for each partner connection.
- Define and retrieve attributes from the IdP adapter, authentication policy contracts, or security token service (STS) token processor to populate an attribute contract directly or use these attributes to look up additional attributes in IdP data stores.
- Define and enforce a set of required attributes needed by SP adapters or STS token generators to interface local systems or applications.
- Set up connections to local data stores.
- Configure specific attribute sources and lookups based on the data stores and map attributes into IdP assertions or into SP adapters or token generators used to interface target applications.
- Selectively mask attribute values recorded in transaction logs.

Attribute contracts

An attribute contract represents an agreement between partners about user attributes sent in a SAML assertion, a JSON web token (JWT), or an OpenID Connect ID token.

The contract is a list of case-sensitive attribute names. Partners must configure attribute contracts to match.

> **Tip**
>
> When privacy is required for sensitive attributes, you can configure PingFederate to mask their values in log files. For more information, see Attribute masking.

For an identity provider (IdP) or an OpenID Provider (OP), the attribute contract defines which attributes PingFederate sends in an assertion, a JWT, or an ID token. While all users authenticate to the partner through this fixed contract, the values used to fulfill the contract might differ from one user to the next. Relying on a combination of different data sources might also fulfill the attribute contract:

- The IdP adapter or security token service (STS) token processor
- An IdP attribute source, which identifies the location of individual attributes in a datastore
- Static text values for some attributes, or text values combined with variables
- Expressions (see Attribute mapping expressions)

For a service provider (SP) or an OpenID Connect Relying Party, the attribute contract defines the attributes PingFederate expects in a SAML assertion, an ID token, or from the UserInfo endpoint at the OP. To pass these attributes to the SP adapter or, for web services, to the SP token generator, configure PingFederate accordingly. For more information, see Managing SP adapters or Managing token generators. In addition, you can configure PingFederate to use attributes to look up additional attributes in local data stores, which often help start a user session or create a local security token for web services. For more information, see Adapter contracts or STS token contracts.

The attribute contract always contains the user identifier `SAML_SUBJECT` in a SAML assertion and `sub` in a JWT or an ID token unless you are using account linking for browser-based single sign-on (SSO). This attribute is automatically included when creating a new contract.

> **Note**
>
> You create attribute contracts on a per-connection basis. For example, if an SP has deployed two session-creation adapters for two separate applications, the IdP connection partner creates a single attribute contract. This single contract supplies all the attributes required by both SP adapters.

Name formats

By agreement with an SP partner, an IdP might specify a format, such as email, associated with the `SAML_SUBJECT`. The SP might further require this information to facilitate handling of the format.

The partner agreement might also include a requirement for the IdP to provide format specifications associated with other attributes.

PingFederate provides a means for an IdP administrator to select from among standard subject, attribute formats, or both, depending on the relevant SAML specifications. An administrator also defines a customized selection of additional attribute formats. For more information, see Setting up an attribute contract.

> **Note**
>
> The designation of formats does not apply to SP administrators. The information about formats remains available in the incoming assertion to an SP application that needs the information for particular processing requirements.

For the WS-Trust IdP configuration, attribute-name formats remain unspecified. If needed however, an administrator might user a special variable in the attribute contract to set the subject-name format. For more information, see Defining an attribute contract for IdP STS. Browser-based SSO attribute contracts also use the same variable, but the feature has deprecated.

STS namespaces

By agreement with an SP partner for a WS-Trust STS connection, an IdP specifies an XML namespace to associate with an attribute, for example, to use claims-based authorization with WIF clients. For more information, see WSC and WSP support. The only attributes that allow specified namespaces belong to a WS-Trust IdP configuration using `SAML 1.1` or `SAML 1.1 for Office 365` as the default token type. For more information, see Defining an attribute contract for IdP STS.

Adapter contracts

An adapter contract represents an agreement between the PingFederate server and an external application.

In concert with the attribute contract between partners, adapter contracts specify the transfer of attributes. Adapter contracts consist of a list of case-sensitive attribute names.

On the identity provider (IdP) side of a federation, an IdP adapter supplies attributes to PingFederate for more information, see Bundled adapters and authenticators and Managing IdP adapters.

On the service provider (SP) side, adapters require adapter contract attributes to start a session with an application. Each security domain requires at least one adapter type. Then, you must configure an adapter instance for each target application. For more information, see Managing SP adapters.

Attributes from the attribute contract fulfill the adapter contracts on the SP side, possibly enhanced with other attributes from local data stores. For example, if the same security context controls several target applications and provides the same set of attributes to start a session for the user, you would deploy an adapter type and configure an adapter instance for each protected application. For more information, see Managing target session mappings.

Extended adapter contract

When PingFederate deploys an adapter type, it creates adapter contracts. Developing these adapters "hard-wires" them to look up or set a specific set of attributes. Attribute requirements might change after deployment. To streamline adjustment of adapter contracts, PingFederate allows an administrator to add additional attributes to the adapter instance through the administrative console, called extended adapter contracts.

STS token contracts

Similar to an adapter contract for broswer-based single sign-on (SSO), A security token service (STS) token-processor or token-generator contract represents an agreement between the PingFederate server and an external application in the context of a web services transaction.

In concert with the attribute contract between partners, token contracts specify the transfer of attributes, consisting of a list of case-sensitive attribute names.

On the identity provider (IdP) side of a federation, PingFederate receives token-processor attributes. For more information, see Token processors and generators and Managing token processors.

On the service provider (SP) side, a token generator requires token-generator contract attributes to pass identify information from the token to the web service client application. Each security domain requires at least one token generator type. Then a token-generator instance must be configured for each target application. For more information, see Managing token generators. If several target applications are controlled by the same security context and can receive the same set of attributes for the user, you would deploy a token generator type and configure a token generator instance for each target application. For more information, see Managing SP token generator mappings.

Extended token generator contract

When PingFederate deploys a token-generator type, it creates token-generator contracts. When developed, these token generators are “hard-wired” to look up or set a specific set of attributes. After deployment, your attribute requirements might change. To streamline adjustment of token-generator contracts, PingFederate allows an administrator to add additional attributes to the token-generator instance through the administrative console. These adjustments are called extended token-generator contracts.

Datastores

Datastores represent external systems that store user attributes and other data.

Once defined, PingFederate configurations retrieve user attributes from datastores for contract fulfillment and token authorization in various use cases. PingFederate configurations write certain records or log messages to datastores.

PingFederate supports a wide variety of database servers and directory servers. As needed, the PingFederate SDK supports the creation of custom drivers for connecting to other types of data repositories, such as flat files or SOAP-connected databases. For more information, see the Javadoc for the `CustomDataSourceDriver` interface, the `SamplePropertiesDataStore.java` file for a sample implementation, and the SDK Developer’s Guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

Related links

- System requirements
- Attribute mapping with multiple data sources

Attribute masking

At runtime PingFederate logs user attributes. To preserve user privacy, you can mask the values of logged attributes.

For more information about log files, see PingFederate log files. PingFederate provides this masking capability at all points where the server logs attributes. These points include:

- Datastore lookup at either the identity provider (IdP) or service provider (SP) site. For more information, see Datastores.
- Retrieval of attributes from an IdP adapter or token processor. For more information, see Setting pseudonym and masking options and Setting attribute masking.
- SP-server processing of incoming attributes based on the single sign-on (SSO) attribute contract. For more information, see Defining an attribute contract.

> **Note**
>
> The SAML Subject ID is not masked; the SAML specifications provide for either pseudonymous account linking or transient identification to support privacy for the Subject ID. For more information, see Account linking.

- SP-server processing of incoming attributes in response to an Attribute Request under X.509 Attribute Sharing Profile (XASP). For more information, see Configuring security policy for Attribute Query.

  For information about XASP, see Attribute Query and XASP.

> **Important**
>
> Many adapter implementations, along with other product extensions, can independently write unmasked attribute values to the PingFederate server log. PingFederate does not control these implementations. If using such a component raises a concern about sensitive attribute values, you can adjust the component's logging threshold in `log4j2.xml` to prevent the recording of attributes.

Token authorization

PingFederate provides an optional configuration known as token authorization to evaluate user attributes as well as other runtime variables, such as authentication context, for authorization purposes.

Token authorization provides a way for administrators to extend access policy directly to many areas, such as browser single sign-on (SSO), security token service (STS), and OAuth events, by conditionally allowing or disallowing the issuance of relevant security tokens such as SAML assertions, STS tokens, OAuth access tokens, or session cookies. The option is also available for extending authorization policy to attribute-query responses, identity provider (IdP) adapter contracts, and authentication policy contracts.

Administrators can configure token authorization using Issuance Criteria windows immediately following the configuration of attribute mapping at all applicable points in the administrative console. See Defining issuance criteria for IdP Browser SSO as an example.

Issuance criteria

The token-authorization configuration consists of rules that evaluate attribute values against selected conditions. Depending on the type of configuration that contains the token-authorization setup, choose from among several sources for the attributes. The sources always consist of all of those available for attribute mapping, including configured data stores and runtime information related to the context of an event. In addition, the sources use values of mapped attributes to provide access to any plain-text mappings or the runtime results of any attribute mapping expressions.

> **Tip**
>
> When more than one condition is configured, all are evaluated and all the conditions must be met at runtime (evaluated as true) for authorization to succeed and processing to continue. In cases where you might need "or" conditions or layered evaluations, you can create one or more attribute mapping expressions.

> **Note**
>
> When authorization fails and a transaction halts, the system passes back a configurable `Error Result` code, potentially to an application layer or as a variable on a PingFederate user-facing template. How this code is interpreted depends on the use case and application integration.

Single and multivalued conditions

Each token-authorization configuration provides a choice of conditions for evaluating attribute values:

- equal to
- equal to (case insensitive)
- equal to DN
- not equal to
- not equal to (case insensitive)
- not equal to DN
- multi-value contains
- multi-value contains (case insensitive)
- multi-value contains DN
- multi-value does not contain
- multi-value does not contain (case insensitive)
- multi-value does not contain DN

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions when you want PingFederate to validate whether one of the attribute values matches or does not match the specified value. Using a single-value condition when an attribute has multiple values causes the criteria to fail consistently.

Related links

- Attribute mapping expressions

User provisioning

PingFederate provides cross-domain user provisioning and account management.

User provisioning is an important aspect of identity federation. When organizations enable SSO for their users, they must ensure that some form of account synchronization is in place. Automated user provisioning features within PingFederate free administrators from having to devise a manual strategy for this.

Provisioning support takes different forms, depending on what role PingFederate plays in an identity federation, and you can configure it either in conjunction with partner SSO connections or separately:

- IdP sites support automatic provisioning and maintaining user accounts at service provider (SP) sites that have implemented the system for cross-domain identity management (SCIM) or at selected software as a service (SaaS) providers. For more information, see the next section, Outbound provisioning for IdPs.

  For information about SCIM, see www.simplecloud.info.

- When PingFederate is configured as a SP, it supports provisioning and managing user accounts and groups for your own organization automatically by using the standard SCIM protocol or by using identity information received during SSO events from SAML assertions. For more information, see Provisioning for SPs.

Outbound provisioning for IdPs

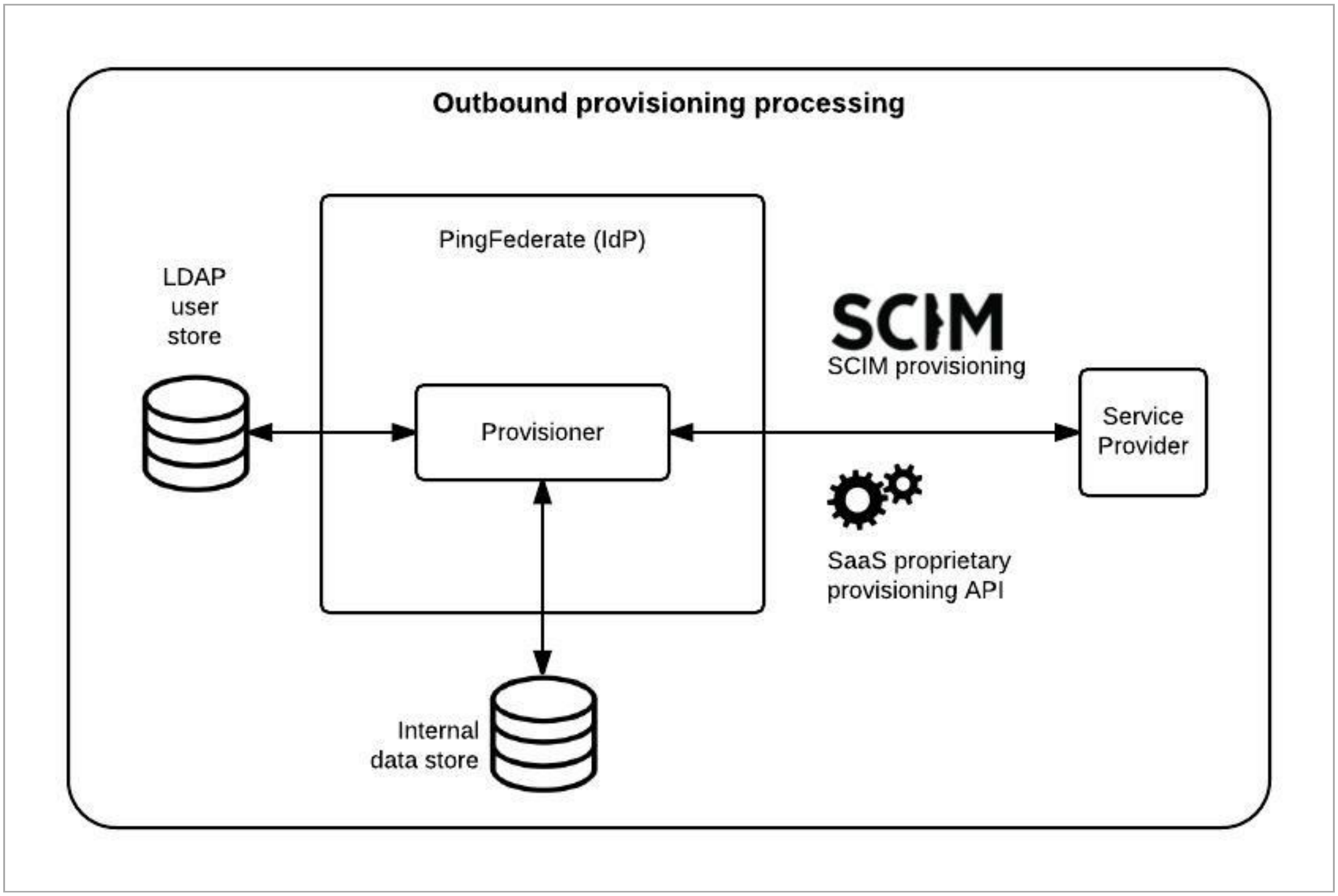
For identity provider (IdP) sites, PingFederate provides built-in automated provisioning and user-account management to system for cross-domain identity management (SCIM)-enabled services providers and to selected software as a service (SaaS) providers through their proprietary provisioning APIs.

User provisioning is an important aspect of identity federation. When organizations enable SSO for their users, they must ensure that some form of account synchronization is in place. Automated user provisioning features within PingFederate free administrators from having to devise a manual strategy for this.

Outbound provisioning also provides an automated means of account disabling or deprovisioning, which might be of key importance to system auditors.

> **Tip**
>
> Support for provisioning for SaaS applications, including quick-connection templates to expedite the configuration effort, is available separately. Contact sales@pingidentity.com for more information.

With outbound provisioning enabled, the PingFederate runtime engine, the provisioner, polls the IdP organization's user store periodically. The server uses a separate database to monitor the state of the user store and keeps user data synchronized between the organization and the target service provider, as illustrated in the following diagram.



LDAP user store

PingFederate provides built-in support for PingDirectory, Microsoft Active Directory, Oracle Unified Directory; pre-configuration of many provisioning settings uses templates. Although Ping Identity has only formally tested these datastores for support, other LDAP datastores will likely work as well.

Internal datastore

PingFederate is tested with Amazon Aurora (MySQL and PostgreSQL), Microsoft SQL Server, Oracle Database, Oracle MySQL, and PostgreSQL as internal provisioning datastores. A demonstration-only, embedded HSQLDB database is installed by default. Scripts to aid setup are in the directory `<pf_install>/pingfederate/server/default/conf/provisioner/sql-scripts`.

Caution

Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Provisioning for SPs

User provisioning is an important aspect of identity federation. When organizations enable single sign-on (SSO) for their users, they must ensure that some form of account synchronization is in place. Automated user provisioning features within PingFederate free administrators from having to devise a manual strategy for this.

When configured as an service provider (SP), PingFederate offers two provisioning options: inbound provisioning or just-in-time (JIT) provisioning.

Inbound provisioning

System for Cross-domain Identity Management (SCIM) inbound provisioning provides support for incoming SCIM messages containing requests to create, read, update, delete, or deactivate user and group records in Microsoft Active Directory datastores or custom user stores through the identity store provisioners. PingFederate supports SCIM attributes in the core schema and custom attributes through a schema extension. Configuring this provisioning feature has two options: by itself or in conjunction with SSO or other connection types.

In effect, inbound provisioning provides an organization with a dedicated SCIM service provider, which routes user-managment requests to an organization's centralized user store. The requests usually originate from trusted applications within an organization, such as a human-resources onboarding software as a service (SaaS) product, or from a trusted partner identity provider (IdP).

Learn more about configuration in Configuring SCIM inbound provisioning.

Learn more about integrating inbound provisioning with custom user stores in Configuring Identity Store Provisioners.

Learn more about application development using PingFederate endpoints for SCIM provisioning in SCIM inbound provisioning endpoints.

Just-in-time provisioning

At an SP site, PingFederate creates and updates local user accounts in an external LDAP directory or Microsoft SQL Server as part of SSO processing, called just-in-time (JIT) provisioning or, formerly, express provisioning. When provisioning requires local accounts, this feature allows SPs to maintain accounts for users who authenticate through IdP partners without having to provision accounts manually.

When configured, the PingFederate SP server writes user information to the local user store using attributes from the incoming SAML assertion. For SAML 2.0 partner connections, supplement assertion attributes with user attributes returned from an attribute query.

PingFederate also updates existing user accounts based on assertions. Using this option, PingFederate adds or overwrites attributes for a local user account each time PingFederate processes SSO for a user.

Learn more about enabling JIT Provisioning in Choosing IdP connection options.

Learn more about configuration in Configuring just-in-time provisioning.

Customer identity and access management

PingFederate empowers administrators to deliver a secure and easy-to-use customer authentication, registration, and profile management solution.

This solution leverages the HTML Form Adapter to offer users the options to authenticate through third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page. The registration and profile management pages are fully customizable and localizable, which allows administrators to present a consistent branding experience based on the needs of the users and the organizations.

Related links

- Customer IAM configuration

Federation hub use cases

Configuring PingFederate as a federation hub accomplishes two primary functions.

Configure PingFederate as a federation hub to:

- Bridge partners using different federation protocols to circumvent partner or application limitations.
- Multiplex a connection for multiple partners to reduce costs and expand use cases.

As a federation hub, PingFederate bridges browser-based single sign-on (SSO) between identity providers and service providers. It stands in the middle of the SSO and single log-out (SLO) flow, acting as the service provider (SP) for the identity providers and as the identity provider (IdP) for the service providers. The four use cases are:

- Bridging an IdP to an SP
- Bridging an IdP to multiple SPs
- Bridging multiple IdPs to an SP
- Bridging multiple IdPs to multiple SPs

PingFederate also supports protocol translation among SAML 1.0, 1.1, 2.0, OpenID Connect, and WS-Federation. For SAML-based connections, this also means it is possible to bridge between various bindings between identity providers and service providers.

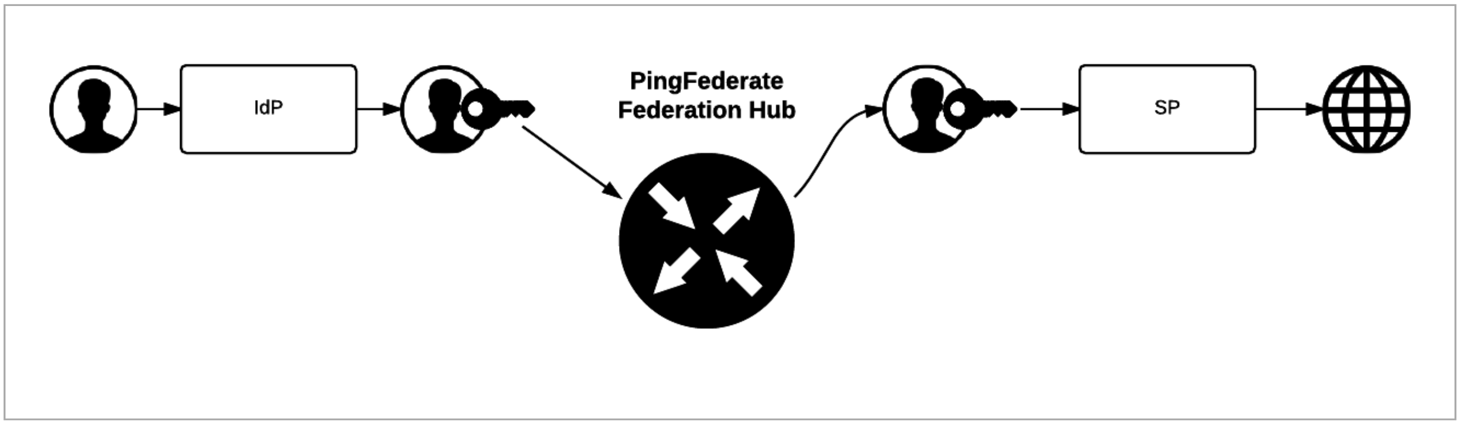
The federation hub capability deploys alongside with other OAuth use cases, IdP connections, SP connections, or any combination of them, to your partners. This flexibility helps in streamlining your federation infrastructure and reducing operating costs.

Bridging an IdP to an SP

PingFederate bridges single sign-on (SSO) and single log-out (SLO) transactions between an identity provider (IdP) and a service provider (SP).

About this task

If you have a legacy IdP system only capable of sending SAML 1.1 assertions through POST and an SP that requires SAML 2.0 assertions through the artifact binding, configuring the federation hub allows PingFederate to consume inbound SAML 1.1 assertions by POST, translate them to SAML 2.0 assertions, and send them through the artifact binding the SP.



Steps

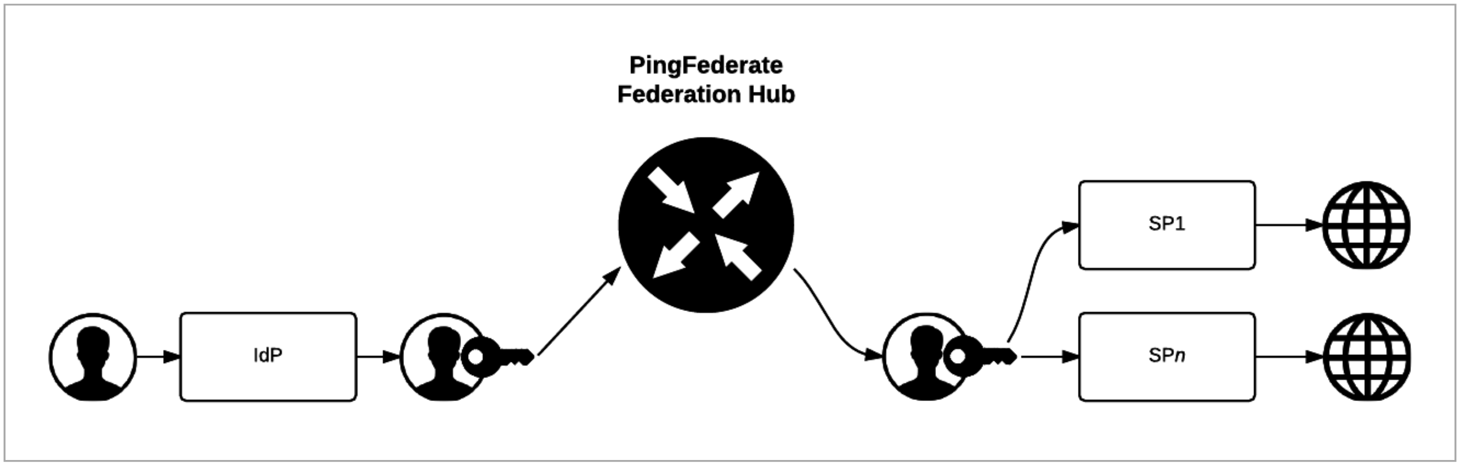
1. Create a contract to bridge the attributes between the IdP and the SP. For more information, see Federation hub and authentication policy contracts.
2. Create an IdP connection between the IdP and PingFederate, the federation hub as the SP, and add the applicable authentication policy contracts to the IdP connection on the **Target Session Mapping** tab.
3. Create an SP connection between PingFederate, the federation hub as the IdP, and the SP and add to the SP connection the corresponding authentication policy contract on the **Authentication Source Mapping** window.
4. Work with the IdP to connect to PingFederate, the federation hub, as the SP.
5. Work with the SP to connect to PingFederate, the federation hub, as the IdP.

Bridging an IdP to multiple SPs

PingFederate bridges single sign-on (SSO) and single log-out (SLO) transactions between an identity provider (IdP) and multiple service providers (SPs).

About this task

For example, your company wants to route federation requests from a recently acquired subsidiary through its federation infrastructure. PingFederate multiplexes one IdP connection to multiple SP connections to the desired SPs. The federation hub consumes assertions from the subsidiary and creates new assertions to the respective SPs.



Steps

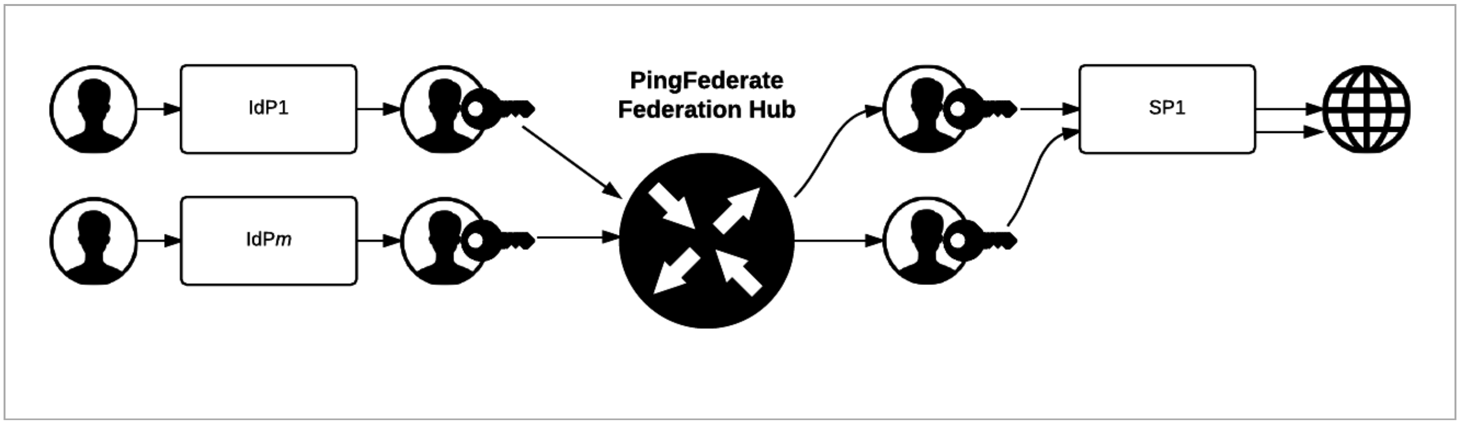
1. For each SP, create a contract to the IdP. For more information, see Federation hub and authentication policy contracts. Because each SP likely requires a unique set of attributes, you will need to create multiple contracts.
2. Create an IdP connection between the IdP and PingFederate, the federation hub as the SP.
3. Add the applicable authentication policy contract(s) to the IdP connection on the **Target Session Mapping** window.
4. For each SP, create an SP connection between PingFederate, the federation hub as the IdP, and the SP.
5. Add the corresponding authentication policy contract to the SP connection on the **Authentication Source Mapping** window.
6. For each SP supporting the SAML IdP-initiated SSO profile, map the expected target resources to the corresponding SP connections on the **Applications > Integration > Target URL Mapping** window.
7. Work with the IdP to connect to PingFederate , the federation hub as the SP.
8. Work with each SP to connect to PingFederate, the federation hub as the IdP.

Bridging multiple IdPs to an SP

With PingFederate, you can bridge single sign-on (SSO) and single log-out (SLO) transactions between multiple identity providers (IdPs) and a service provider (SP).

About this task

For example, you are tasked to provide federated access to resources on Microsoft SharePoint for various business partners. With PingFederate, you can multiplex one SP connection, to SharePoint, to multiple IdP connections for all your business partners. The federation hub can also, as needed, translate SAML assertions from the business partners to WS-Federation security tokens and send them over to SharePoint.



Steps

1. Create a contract to bridge the attributes between the IdPs and the SP. For more information, see Federation hub and authentication policy contracts.

   You likely need only one contract unless the SP requires a different set of attributes from each IdP.
2. For each IdP, create an IdP connection between the IdP and PingFederate, the federation hub as the SP.

3. On the **Target Session Mapping** window, add the applicable authentication policy contracts to the IdP connection.

4. On the **Selectors** window, configure an authentication selector. For example, see an instance of the Identifier First Adapter to map each IdP to the corresponding IdP connection in an authentication policy.

5. Create an SP connection between PingFederate, the federation hub as the IdP, and the SP.

6. Add the corresponding authentication policy contract to the SP connection on the **Authentication Source Mapping** window.

> **Important**
>
> PingFederate includes the Entity ID of the original IdP (`Authenticating Authority`) in SAML 2.0 assertions so that the SP can determine the original issuer of the assertions. This is especially important when bridging multiple IdPs to one SP—the SP should take the information about the original issuer into consideration before granting access to protected resources.
> For SAML 1.x assertions and WS-Federation security tokens, you can add an attribute on the **Attribute Contract** window and then map **Context: Authenticating Authority** as the attribute value on the **Attribute Contract Fulfillment** window.
> For information about `Authenticating Authority` , see section in the SAML 2.0 specification.

> **Note**
>
> If the SP does not take action based on `Authenticating Authority` , depending on the attributes from the IdPs, you can define validation rules on the **Issuance Criteria** window to protect against user impersonation between IdPs.

7. Work with each IdP to connect to PingFederate, the federation hub as the SP.

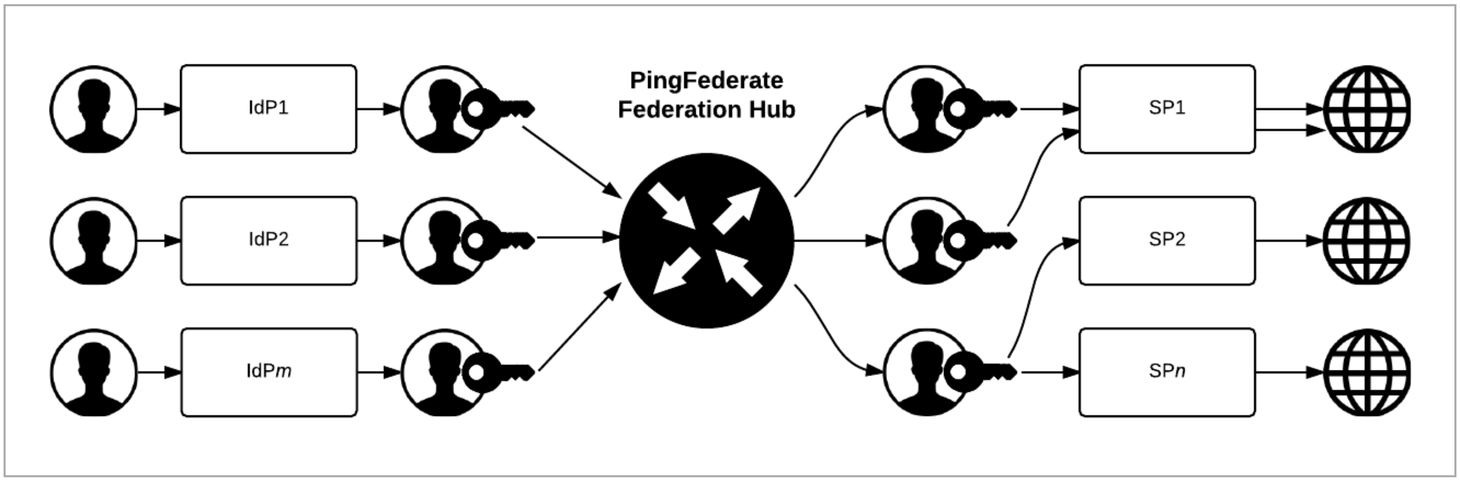
8. Work with the SP to connect to PingFederate, the federation hub as the IdP.

Bridging multiple IdPs to multiple SPs

PingFederate can bridge single sign-on (SSO) and single log-out (SLO) transactions between multiple identity providers (IdPs) and service providers (SPs).

About this task

This PingFederate federation hub use case is a combination of Bridging an IdP to multiple SPs and Bridging multiple IdPs to an SP.



Steps

1. Create multiple contracts to bridge the attributes between the IdPs and the SPs. For more information, see Federation hub and authentication policy contracts.

2. For each identity provider, create an IdP connection between the IdP and PingFederate, the federation hub as the SP.

3. Add the applicable authentication policy contracts to the IdP connection in the **Target Session Mapping** window.

4. In the **Selectors** window, configure an authentication selector to map each IdP to the corresponding IdP connection in an authentication policy. For example, see an instance of the Identifier First Adapter.

5. For each service provider, create an SP connection between PingFederate, the federation hub as the IdP, and the SP.

6. Add the corresponding authentication policy contract to the SP connection in the **Authentication Source Mapping** window.

   Important

   PingFederate includes the Entity ID of the original IdP, `Authenticating Authority`, in SAML 2.0 assertions so that the SP can determine the original issuer of the assertions. This is especially important when bridging multiple IdPs to one SP—the SP should take the information about the original issuer into consideration before granting access to protected resources.
   In the **Attribute Contract** window, add an attribute for SAML 1.x assertions and WS-Federation security tokens. Then, in the **Attribute Contract Fulfillment**, map **Context: Authenticating Authority** as the attribute value.
   For information about `Authenticating Authority`, see section in the SAML 2.0 specification.

   Note

   If the SP does not take action based on `Authenticating Authority`, in the **Issuance Criteria** window, you can define validation rules to protect against user impersonation between IdPs.

7. For each SP supporting the SAML IdP-initiated SSO profile, map the expected target resources to the corresponding SP connections in the **Applications > Integration > Target URL Mapping** window.

8. Work with each IdP to connect to the federation hub as the SP.

9. Work with each SP to connect to the federation hub as the IdP.

Federation hub and authentication policy contracts

PingFederate uses two connections to bridge an identity provider (IdP) to a service provider (SP). It fuses these two connections together by using an authentication policy contract as the medium to carry user attributes from the IdP to the SP.

The two connections are:

- An IdP connection where end users authenticate and PingFederate, the federation hub, is the SP
- An SP connection to the target application where PingFederate, the federation hub, is the IdP

Each authentication policy contract comes with one default attribute, `subject`. You can extend the contract to include additional attributes as needed. In most federation hub use cases, you configure PingFederate to pull attribute values from inbound assertions into the authentication policy contract in an IdP connection and to push those values from the authentication policy contract into the outbound assertions through an SP connection. For advanced use cases, you can configure the IdP connections, SP connections, or both, to look up values from multiple data store instances.

PingFederate can use OGNL expressions to customize authentication request and response messages from IdP connections before sending them to SPs. To customize messages, the expressions can use the variables *#FedHubIncomingAuthnResponse*, *#FedHubIncomingAuthnRequest*, and *#FedHubOutgoingAuthnRequest* in an SP connection when an authentication policy is used to select the IdP connection. For more information, see Customizing assertions and authentication requests and Message types and available variables.

When bridging one IdP to one SP, you must create one authentication policy contract and associate the contract with both the IdP connection and the SP connection.

When bridging one IdP to multiple SPs, you need to create an authentication policy contract per SP because each SP likely requires a different set of attributes. In addition, you need to map all the authentication policy contracts into the IdP connection and add the respective authentication policy contracts to each SP connection to the SP.

When bridging multiple IdPs to one SP, you likely need only one contract unless the SP requires a different set of attributes from each IdP. In addition, you need to add the authentication policy contract to the SP connection and the applicable IdP connections.

You can manage authentication policy contracts on the **Policy Contracts** window (**Authentication** > **Policies** > **Policy Contracts**).

Federation hub and virtual server IDs

PingFederate manages the federation hub differently based on how the server provider (SP) connection uses virtual server IDs.

PingFederate uses two connections to bridge an identity provider (IdP) to a SP:

- An IdP connection where end users authenticate and PingFederate, the federation hub, is the SP
- An SP connection to the target application where PingFederate, the federation hub, is the IdP

Generally speaking, PingFederate consumes assertions from the IdP through the IdP connection and generates new assertions to the SP through the SP connection.

If the SP connection does not use a virtual server ID, the issuer of the assertions to the SP is the ID defined for the protocol between PingFederate, the federation hub as the IdP, and the SP.

If the SP connection uses multiple virtual server IDs for the purpose of connecting to multiple environments serviced by the same partner using one connection, PingFederate automatically retains information about AuthnRequest messages sent to the virtual server ID specific endpoint for SP-initiated single sign-on (SSO). When the IdP returns the corresponding assertions to PingFederate as the SP, PingFederate retrieves the preserved information and uses that specific virtual server ID as the issuer in assertions sent to the SP. For IdP-initiated SSO, the issuer of the assertions to the SP is the default virtual server ID.

Federation planning checklist

An essential first step in establishing an identity federation involves discussions and agreements between you and your connection partners. The sections below comprise a partial checklist of items that should be coordinated before you deploy PingFederate.

Standards and specifications

Choose which federation protocols your deployment will support. For SAML single sign-on (SSO) configurations, decide which profiles and bindings will be used. For more information, see Supported Standards.

Signing and validation

Decide which SAML messages—assertions, responses, requests—will be digitally signed and how the messages will be verified by your federation partner. If messages are signed, decide how certificates will be exchanged, for example, secure email. For more information, see Security infrastructure.

Back-channel security

Determine what type of SOAP channel authentication will be used: Basic or SSL/TLS. If SSL/TLS is used, determine whether server-only or both server and client certificates will be needed and how they will be managed. Also decide what level of security will be required for connections to back-end datastores or identity management systems.

Trusted certificate management

Determine whether both partners are using SSL/TLS runtime certificates, signing certificates, or both that have been signed by a major certificate authority (CA). If self-signed certificates or nonstandard CAs are used, the signed certificates must be exchanged and imported into Trusted Certificate stores. Also, determine whether you want to adopt a trust model that uses embedded certificates. For more information, see Digital signing policy coordination.

Deployment

Decide how PingFederate fits into your existing network. Also, determine whether high-availability, failover options, or both, are required. For more information, see the PingFederate Server Clustering Guide.

Federation server identification

Determine how you and your partners will identify your respective federation deployments. Under federation standards, both the sender, the identity provider (IdP), and the receiver, service provider (SP), of an assertion must be uniquely identified within the identity federation. For more information, see Configuration data exchange.

With PingFederate, you define a unique ID for each supported protocol. For more information, see Specifying federation information. Optionally, you can also use a list of multiple virtual server IDs on a connection-by-connection basis. For more information, see Multiple virtual server IDs.

Tip

PingFederate also provides for virtual host names, which differ from virtual server IDs but are not mutually exclusive; they are intended for use when your network configuration is such that you receive federation messages under more than one domain name. For more information, see Configuring virtual host names.

Server clock synchronization

Ensure that both the SP and IdP server clocks are synchronized. SAML messages and security token service (STS) tokens provide a time window that allows for small synchronization differentials. However, wide disparities will result in assertion or request time-outs.

User data stores

Identify the type of datastore that contains user data when needed. For more information, see Datastores.

Web application and session integration

Decide how PingFederate as an IdP receives subject identity information, either from an STS token or a user session.

For an SP, decide how PingFederate will forward user identity information to the destination web application or system to start a session. For more information, see Bundled adapters and authenticators and Token processors and generators.

Transaction logging

PingFederate provides basic transaction logging and monitoring. Decide whether transaction logging should be integrated with a systems management application and whether you have regulatory compliance requirements that affect your logging processes. For more information, see PingFederate log files.

Identity mapping

For browser-based SSO, decide whether you will use PingFederate to link accounts on your respective systems using a persistent name identifier, or whether you will use account mapping. For more information, see Identity mapping.

Attribute contract agreement

If your federation partnership will not use account linking, or will not use it exclusively, then you and your partner must agree on a set of attributes that the IdP will send in an assertion for either SSO or web service access. For more information, see Attribute contracts

Metadata exchange

If you are using SAML, decide whether you will use the metadata standard to exchange XML files containing configuration information. PingFederate makes it easy to use this protocol, which provides a significant shortcut to setting up your partner connections.

Multiple virtual server IDs

Virtual server IDs provide more configuration flexibility in cases where you need to identify your server differently when connecting to a partner in one connection for multiple environments or in multiple connections where the partner also supports multiple federation IDs.

Connecting to a partner in one connection

This is a use case where you need to connect to multiple environments serviced by the same partner using one federation ID, multiplexing one service provider (SP) connection to access multiple subdomain accounts in Microsoft Office 365.

Suppose both the marketing and the engineering departments of contoso.com, the identity provider (IdP), have their own departmental subdomains, marketing.contoso.com and engineering.contoso.com. They are both registered in Office 365, the SP, under the parent domain, contoso.com.

To include both marketing.contoso.com and engineering.contoso.com as the virtual server IDs in the Office 365 SP connection, configure the PingFederate IdP server. Each virtual server ID has its own set of protocol endpoints obtained in the connection metadata. For more information, see Metadata export and System-services endpoints.

After providing the protocol endpoints information to Office 365, when Office 365 sends login requests to PingFederate, PingFederate picks the correct IdP adapter to authenticate the end users based on the virtual server ID in the requests.

For each successful login, PingFederate builds an assertion with the issuer set to the corresponding virtual server ID. When Office 365 receives the assertion, it creates the end user session with the right subdomain settings based on the issuer value in the assertion.

Connecting to a partner in multiple connections

In this use case, you connect to your partner in multiple connections. In each connection, you identify yourself and your partner differently.

For example, you as the SP provide separate environments for the end users based on their regions. Your IdP operates in two regions, Europe (EU) and North America (NA). Their federation IDs are `eu.idp.local` and `na.idp.local` respectively.

In the PingFederate SP server, you can create two IdP connections to federate identities for end users from both regions as follows.

	Partner’s federation ID	Your virtual server ID
IdP connection #1	`eu.idp.local`	`idp-eu.sp.tld`
IdP connection #2	`na.idp.local`	`idp-na.sp.tld`

Based on the issuer (the partner’s federation ID) and the audience values (your virtual server ID), PingFederate determines at runtime which IdP connection the assertion is intended for, validates as per the connection settings, and passes attribute values to the SP adapter to create the end-user session.

Working with multiple virtual server IDs

You can assign virtual server IDs either as an IdP during configuration of an SP connection or as an SP configuring an IdP connection for both Browser single sign-on (SSO) Profiles and WS-Trust security token service (STS) for access to identity-enabled web services. For more information, see Identifying the SP and Identifying the partner.

If a connection has only one virtual server ID, it becomes the default virtual server ID for the connection. If the list contains several entries, you must specify one of them as the default virtual server ID for that connection. The connection uses the default virtual server ID when a request does not include virtual server ID information. For more information, see IdP endpoints for an IdP or SP endpoints for an SP.

In a connection with multiple virtual server IDs, you can restrict each adapter added to the connection to certain virtual server IDs to enhance the end-user experience. For more information, see Restricting an authentication source to certain virtual server IDs and Restricting a target session to certain virtual server IDs.

> **Tip**
>
> Restrict each token processor or token generator added to a WS-Trust STS SP connection or IdP connection. For more information, see Restricting a token processor to certain virtual server IDs or Restricting a token generator to certain virtual server IDs.

> **Important**
>
> To protect against unauthorized access, configure Issuance Criteria to verify virtual server ID in conjunction with other conditions, such as group membership information. For more information, see Defining issuance criteria for IdP Browser SSO or Defining issuance criteria for SP Browser SSO.

Configuration data exchange

If your partner’s deployment does not produce or consume a metadata file that conforms to SAML metadata specifications, you might need to exchange connection information manually. If the deployment does not use metadata, some common configuration details must be exchanged.

Identity provider (IdP) to service provider (SP)

If you are the IdP, your SP partner will need some or all of the following connection information, depending upon which profiles and bindings you configure:

- Unique ID—Identifies the IdP that issues an assertion or other SAML message. For SAML 2.0, the ID is the IdP entity ID; for SAML 1.x, it is the IdP issuer; for WS-Federation, it is the IdP realm.

  PingFederate also supports the optional use of virtual IDs. For more information, see Federation planning checklist.
- SOAP artifact resolution URL—The endpoint your site uses to receive an SP’s SOAP requests when the artifact binding is used.
- Single logout (SLO) service URL—The destination of SLO request messages.
- Single sign-on (SSO) service URL—The endpoint where you receive and process assertions.

SP to IdP

If you are the SP, your IdP partner will need some or all of the following connection information depending upon which profiles and bindings you configure:

- Unique ID—Identifies the SP. For SAML 2.0, the ID is the entity ID; for SAML 1.x, it is the SP’s audience; for WS-Federation, it is the SP’s realm.

  PingFederate also supports the optional use of virtual IDs. For more information, see Federation planning checklist.
- SOAP artifact resolution service URL—The endpoint to use for SOAP requests when the artifact binding is used.
- Single logout service URL (SAML 2.0)—The destination of SLO request messages.

- Assertion consumer service URL—The location where the SP receives assertions.
- Target URLs—The URLs for the protected resources that a user is trying to access.

Mutual settings between partiesThe parties must mutually determine the settings. These settings might include:

- Attributes—User information sent in an assertion. For more information, see User attributes.
- Signing certificates—SAML and WS-Federation protocols specify a number of conditions built into the PingFederate connection-setup windows that might or might not require digital signatures.
- SOAP connection type and authentication style—For SAML connections using the back channel, such as artifact binding, HTTP Basic authentication, SSL client certificate authentication, digital signatures, or some combination of the three is required. You and your partner must exchange the necessary credentials, certificates, and signing keys.

Installing and uninstalling PingFederate

PingFederate operates as a standalone server based on Java EE application server technology. This section shows you how to properly install PingFederate.

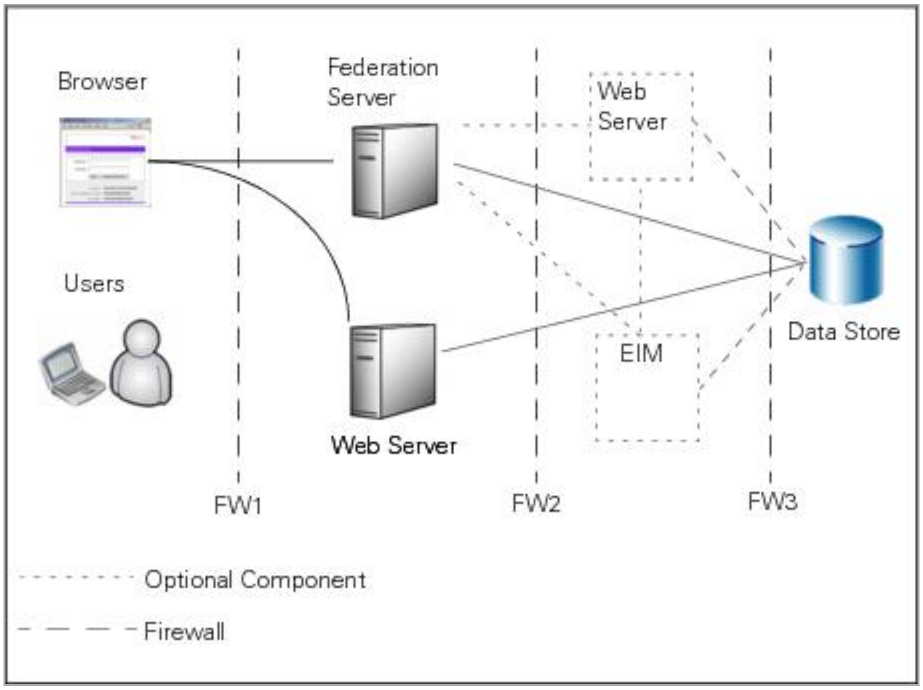
A new installation involves:

- Determining the deployment architecture
- Reviewing system and port requirements
- Installing a Java runtime environment
- Installing PingFederate
- Completing the Initial Setup wizard

Deployment options

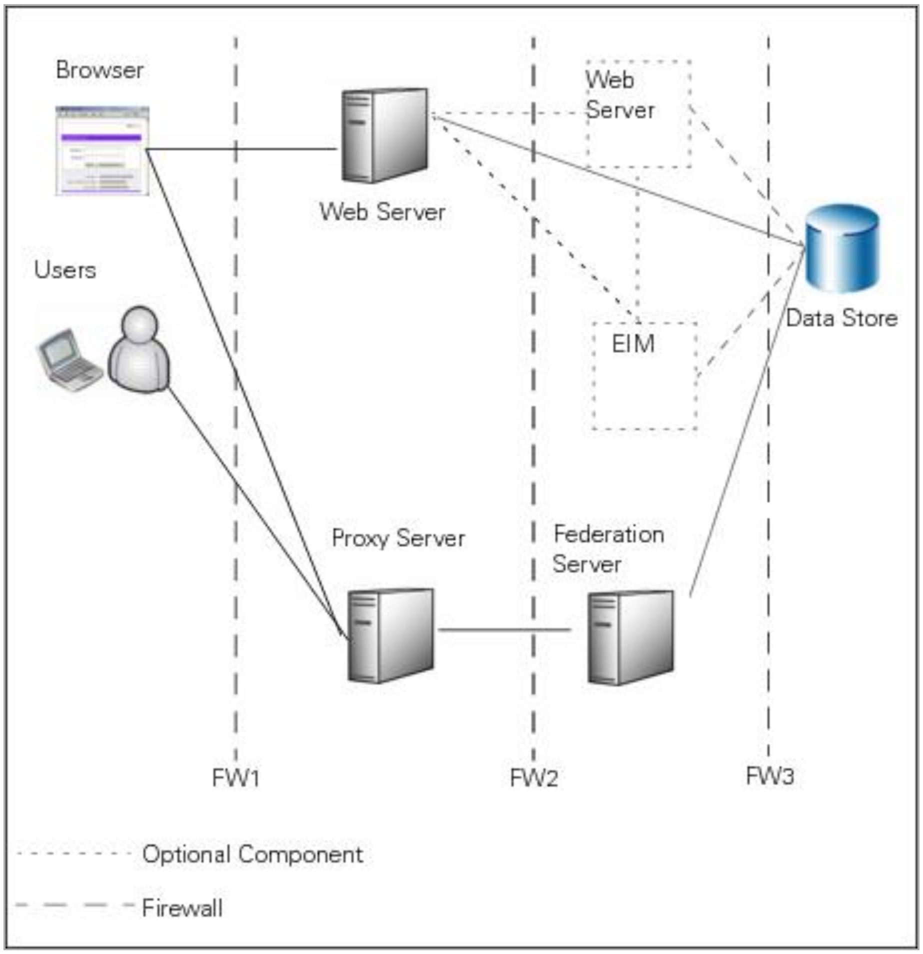
Depending on your needs and infrastructure capabilities, you can choose a standalone or proxy configuration. For information about configuring proxy settings, see Configuring incoming proxy settings and Configuring forward proxy server settings.

The following diagram illustrates a standalone PingFederate deployment in a DMZ.



In this configuration, the users access PingFederate through a web application server, an enterprise identity management (EIM) system, or both. PingFederate then retrieves information from a datastore to use in processing the transaction.

You can also deploy PingFederate with a proxy server. The following diagram depicts a proxy-server configuration in which users and web browsers access the proxy. The proxy then communicates with PingFederate to request single sign-on (SSO).



Related links

- Upgrading PingFederate
- Uninstalling PingFederate

System requirements

PingFederate has the following recommended system versions and requirements.

Operating systems and virtualization

> **Note**
>
> PingFederate is tested with default configurations of operating-system components. If your organization customizes implementations or installs third-party plug-ins, deployment efforts might affect the PingFederate server.

Component	Supported versions
Operating systems	• Amazon Linux 2 and 2022 • Canonical Ubuntu 18.04 LTS and 20.04 LTS • Microsoft Windows Server 2012 R2, 2016, 2019, and 2022 • Oracle Linux (Red Hat Compatible Kernel) 7.9 and 8.6 • Red Hat Enterprise Linux ES 7.9, 8.6, and 9.0 • SUSE Linux Enterprise 12 SP5 and 15 SP4 ⓘ **Note** If you have a Windows Server 2012 R2 environment, you should upgrade to a later version. Learn more, including the end of support for Windows Server 2012 R2 in July 2023, in Upgrade considerations introduced in PingFederate 10.x.
Docker support	• Docker version: 24.0.2 View the PingFederate Docker image on DockerHub. Visit Ping Identity's DevOps documentation for more information. Note that only the PingFederate software is licensed under Ping Identity's end user license agreement, and any other software components contained within the image are licensed solely under the terms of the applicable open source or third-party license. ⓘ **Note** Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance or interoperability of any virtualization software with its products.
Virtualization	Although Ping Identity does not qualify or recommend any specific virtual-machine (VM) or container products other than those already specified, PingFederate has run well on several, including Hyper-V, VMWare, and Xen. ⓘ **Note** The list of products is provided for example purposes only. We view all products in this category equally. Ping Identity accepts no responsibility for the performance of any specific virtualization software and in no way guarantees the performance, interoperability, or both of any VM or container software with its products.

Java environment

- Amazon Corretto 8, 11, and 17
- OpenJDK 11 and 17
- Oracle Java SE Development Kit 11 LTS and 17 LTS
- Oracle Java SE Runtime Environment (Server JRE) 8

Note

Ping Identity Java Support Policy applies. Learn more in Java Support Policy in the Ping Identity Knowledge Base.

Important

PingFederate does not support any JDK 11 version prior to 11.0.4 due to an error covered in the Oracle Java Bug Database.

Browsers

Server	Supported browsers
Runtime server	• Apple Safari • Google Chrome • Microsoft Edge • Mozilla Firefox • Apple iOS 16 (Safari) • Google Android 13 (Chrome)
Administrative server	• Google Chrome • Microsoft Edge • Mozilla Firefox

TLS protocol

Runtime server and administrative server

- TLS 1.2 and 1.3

Datastore integration

Functionality	Supported versions
User-attribute lookup	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Amazon DynamoDB • Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) • Aurora PostgreSQL (compatible with PostgreSQL 14.3) • Azure SQL Managed Instance • Microsoft Active Directory 2012 R2 and 2016 • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Database 12c Release 2 and 19c • Oracle MySQL 8.0 • Oracle Unified Directory 12c • PostgreSQL 11 and 13 • Custom implementation through the PingFederate SDK
SaaS or SCIM outbound provisioning	***Provisioning channel data source*** • PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Microsoft Active Directory 2012 R2 and 2016 • Oracle Unified Directory 12c ***Provisioning internal datastore*** • Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) • Aurora PostgreSQL (compatible with PostgreSQL 14.3) • Azure SQL Managed Instance • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Database 12c Release 2 and 19c • Oracle MySQL 8.0 • PostgreSQL 11 and 13
SCIM inbound provisioning	• Microsoft Active Directory 2012 R2 and 2016 • Custom implementation through the PingFederate SDK
Just-in-time (JIT) inbound provisioning	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Azure SQL Managed Instance • Microsoft Active Directory 2012 R2 and 2016 • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Unified Directory 12c

Functionality	Supported versions
Account linking	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) • Aurora PostgreSQL (compatible with PostgreSQL 14.3) • Azure SQL Managed Instance • Microsoft Active Directory 2012 R2 and 2016 • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Database 12c Release 2 and 19c • Oracle MySQL 8.0 • Oracle Unified Directory 12c • PostgreSQL 11 and 13
OAuth client configuration and persistent grants	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Amazon DynamoDB • Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) • Aurora PostgreSQL (compatible with PostgreSQL 14.3) • Azure SQL Managed Instance • Microsoft Active Directory 2012 R2 and 2016 • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Database 12c Release 2 and 19c • Oracle MySQL 8.0 • Oracle Unified Directory 12c • PostgreSQL 11 and 13 • Custom implementation through the PingFederate SDK
Registration and profile management of local identities	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3
Persistent authentication sessions	• PingDirectory 8.3, 9.0, 9.1, 9.2, 9.3 • Amazon DynamoDB • Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23) • Aurora PostgreSQL (compatible with PostgreSQL 14.3) • Azure SQL Managed Instance • Microsoft SQL Server 2016 SP2, 2017, and 2019 • Oracle Database 12c Release 2 and 19c • Oracle MySQL 8.0 • PostgreSQL 11 and 13 • Custom implementation through the PingFederate SDK

Note

If you have Microsoft Active Directory environments on the 2012 R2 functional level, you should upgrade to a later version. Learn more, including the end of support for the 2012 R2 functional level in July 2023, in Upgrade considerations introduced in PingFederate 10.x.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers.

Secret manager (optional)

CyberArk Credential Provider 12

Hardware security modules (optional)

> **Note**
>
> When integrating with an AWS CloudHSM hardware security module (HSM), Thales Luna HSM, or Entrust nShield Connect HSM, you must deploy with an Oracle Server Java Runtime Environment (JRE), OpenJDK, or Amazon Corretto distribution of Java 8 or 11.

Hardware security module	Supported versions
AWS CloudHSM	• Client software version: 5.9.0 PingFederate must be deployed on one of the Linux or Windows operating systems supported by both AWS CloudHSM and PingFederate.
Entrust nShield Connect HSMs	• Host and Firmware version: 12.80.4 • Client driver version: 12.80.4
Thales Luna Cloud HSM Services and Luna Network HSM	• Luna HSM Client 10.5 Learn more about the Luna HSM Client, including compatible HSMs, HSM firmware, appliance software, and client software, in the Luna Cloud HSM Service Client Guide and the Luna Network HSM Documentation Archive.

Minimum hardware requirements

- Multi-core Intel Xeon processor or higher

  We recommend a minimum of four processing cores distributed across any number of CPUs.

- 4 GB of RAM

  Step 1.5 GB available to PingFederate

- 1 GB of available hard drive space

> **Note**
>
> Although it's possible to run PingFederate on less powerful hardware, the guidelines provided accommodate disk space for default logging, auditing profiles, and CPU resources for a moderate level of concurrent request processing.

Compatible database drivers

PingFederate is compatible with the following vendor-specific Java database connectivity (JDBC) drivers.

Database server	Driver information

Amazon Aurora MySQL 3.02.0 (compatible with MySQL 8.0.23)	***Driver version information***
	mysql-connector-java version 8.0.33
	Driver class
	`com.mysql.cj.jdbc.Driver`
	JDBC URL
	`jdbc:mysql://databaseservername/databasename`
	Database location
	Regional
	Database features
	One writer and multiple readers

Amazon Aurora PostgreSQL (compatible with PostgreSQL 14.3)	***Driver version information***
	postgresql version 42.6.0
	Driver class
	`org.postgresql.Driver`
	JDBC URL
	`jdbc:postgresql://databaseservername/databasename`
	Database features
	One writer and multiple readers

Microsoft SQL Server 2016 SP2, 2017, 2019, and 2022
Microsoft Azure SQL Managed Instance
	Driver version information
	sqljdbc version 12.4.2
	Driver class
	`com.microsoft.sqlserver.jdbc.SQLServerDriver`
	JDBC URL
	`jdbc:sqlserver://databaseservername;databaseName=databasename`

Oracle Database 12c Release 2 and 19c	***Driver version information***
	ojdbc8 version 23.3.0.23
	Driver class
	`oracle.jdbc.OracleDriver`
	JDBC URL
	`jdbc:oracle:thin:@databaseservername:databasename`

Database server	Driver information

Oracle MySQL 8.0	***Driver version information***
	mysql-connector-java version 8.2.0
	Driver class
	`com.mysql.cj.jdbc.Driver`
	JDBC URL
	`jdbc:mysql://databaseservername/databasename`

PostgreSQL 11 and 13	***Driver version information***
	postgresql version 42.7.0
	Driver class
	`org.postgresql.Driver`
	JDBC URL
	`jdbc:postgresql://databaseservername/databasename`

For additional information about these drivers, contact the respective vendors.

Port requirements

The following table summarizes the ports and protocols that PingFederate uses to communicate with external components. This information provides guidance for firewall administrators to ensure the correct ports are available across network segments.

> **Note**
>
> Direction refers to the direction of the initial requests relative to PingFederate. Inbound refers to requests PingFederate receives from external components. Outbound refers to requests PingFederate sends to external components.

PingFederate required ports and protocols

Service	Protocol, direction, transport, default port	Source	Destination	Description
Administrative console	HTTPS, inbound, TCP, 9999	Browsers accessing the administrative console, REST calls to the administrative application programming interface (API), web service calls to the Connection Management Service. Applicable to the console node in a clustered PingFederate environment.	Administrative node	Used for incoming requests to the administrative console. Configurable in the `run.properties` file.
Administrative console	HTTPS, outbound, TCP, 443	Administrator accessing online documentation. Applicable to the console node in a clustered PingFederate environment.	docs.pingidentity.com	Used for accessing online documentation from the administrative console.

Service	Protocol, direction, transport, default port	Source	Destination	Description
Runtime engine	HTTPS, inbound, TCP, 9031 (and 9032 if configured)	Browsers accessing the runtime server for single sign-on (SSO) or single logout (SLO); web service calls to the SSO Directory Service; REST calls to the OAuth Client Management Service, the OAuth Access Grant Management Service, the Persistent Grant Management API, and the Session Revocation API. Applicable to all runtime engine nodes in a clustered PingFederate environment.	Runtime engine nodes	Used for incoming requests to the runtime engine. Configurable in the `run.properties` file.
Cluster traffic	JGroups, inbound, TCP, 7600	PingFederate peer servers in a clustered PingFederate environment.	Administrative node and runtime engine nodes	Used for communications between engine nodes in a cluster when the transport mode for cluster traffic is set to TCP (the default behavior). Configurable in the `run.properties` file.
Cluster traffic	JGroups, inbound, TCP, 7700	PingFederate peer servers in a clustered PingFederate environment.	Administrative node and runtime engine nodes	Used by other nodes in the cluster as part of the cluster's failure-detection mechanism when the transport mode for cluster traffic is set to TCP (the default behavior). Configurable in the `run.properties` file.

Service	Protocol, direction, transport, default port	Source	Destination	Description
PingOne connections (if configured)	HTTPS, outbound, TCP, 443	All nodes	pingone.com	The administrative node uses PingOne APIs to create connections to PingOne. Engine nodes use PingOne APIs to obtain access tokens and call PingOne services.
PingOne for Enterprise integration (if configured)	HTTPS and secure WebSocket, TCP, 443	PingFederate Applicable to the console node in a clustered PingFederate environment.	pingone.com	Used for communications between PingFederate and PingOne for Enterprise for establishing and maintaining a managed SP connection to PingOne for Enterprise, monitoring of PingFederate from the PingOne admin portal, authenticating end users against the PingOne for Enterprise Directory.
Cluster traffic (if configured)	JGroups, outbound, TCP, 443	PingFederate peer servers in a clustered PingFederate environment.	Amazon Simple Storage Service (Amazon S3) or an OpenStack Swift server	Used by all nodes when the optional dynamic discovery mechanism is enabled.
Cluster traffic	JGroups, inbound,UDP, 7601	PingFederate peer servers in a clustered PingFederate environment.	Administrative node and runtime engine nodes	Used for communications between engine nodes in a cluster when the transport mode for cluster traffic is set to UDP. By default, the transport mode is TCP. Configurable in the `run.properties` file.
Active Directory domains/ Kerberos realms (if configured)	Kerberos, outbound, TCP or UDP, 88	PingFederate	Windows domain controllers	Used for communications between PingFederate and Windows domain controllers for the purpose of Kerberos authentication.
reCAPTCHA (if configured)	HTTPS, outbound, TCP, 443	PingFederate	www.google.com/ recaptcha/api/site verify	Used by the HTML Form Adapter when invisible reCAPTCHA from Google is enabled to prevent automated attacks.
Administration notification	SMTP, outbound, TCP, 25 (465 if SMTPS)	All nodes	SMTP server	Used to send notification messages for various events. For more information, see Runtime notifications.

Note

For PingID integration, see PingID required domains, URLs, and ports.
Depending on the integration kits deployed and the connecting third-party systems, such as email server or SMS service provider, additional ports might be required.

Installing Java

PingFederate requires a Java Runtime Environment (JRE) to be installed on your server.

About this task

PingFederate has been tested in the following Java environments:

- Amazon Corretto 8, 11, and 17
- OpenJDK 11 and 17
- Oracle Java SE Development Kit 11 LTS and 17 LTS
- Oracle Java SE Runtime Environment (Server JRE) 8

Note

Ping Identity Java Support Policy applies. Learn more in Java Support Policy in the Ping Identity Knowledge Base.

Important

PingFederate does not support any JDK 11 version prior to 11.0.4 due to an error covered in the Oracle Java Bug Database.

Important

Because of the import restrictions of some countries, Oracle Server Java SE Runtime Environment (JRE) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, enable the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy. For more information, see the Java 8 release notes in Oracle's documentation.
For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide in Oracle's documentation.

Steps

1. Download and install a Java runtime.
2. Set the *JAVA_HOME* environment variable to the Java installation directory path and add its `bin` directory to the *PATH* environment variable.

   Example:

   ```
   JAVA_HOME=C:\Program Files\Java\jdk-17
   PATH=%JAVA_HOME%\bin
   ```

ⓘ **Note**

If you intend to use the PingFederate installer for Windows or run PingFederate as a service, you must set the *JAVA_HOME* environment variable and modify the *PATH* environment variable at the system level. If you are not using the PingFederate installer or running PingFederate as a service, you can set the variables at either the system or user level.

Caution

When running PingFederate for Windows, switching the Java version from 8 to 11 (or the reverse) prevents the service from running, and you won't be able to start PingFederate. The problem occurs because garbage collection logging configuration arguments that are used by Java 8 are incompatible with those used by Java 11. To change Java versions:

1. Run `<pf_install>\pingfederate\sbin\win-x86-64\uninstall-service.bat` to de-register the PingFederate service.
2. Install the new Java version and update the *JAVA_HOME* and *PATH* environment variables.
3. Run `<pf_install>\pingfederate\sbin\win-x86-64\install-service.bat` to register the PingFederate service.

Installing PingFederate

You can install PingFederate on Windows and Linux operating systems.

Install PingFederate using the following methods:

- Install PingFederate on a Windows system by running the installer for Windows or by extracting the distribution `.zip` archive. Using the installer for Windows is the preferred method.
- Install PingFederate on a Linux system by extracting the distribution `.zip` archive.

ⓘ **Note**

This documentation refers to the installation directory path where the `pingfederate` directory is located as *<pf_install>*. For example, `<pf_install>/pingfederate/bin`.

Important

To avoid future problems with automated upgrades, do not rename the installed `pingfederate` directory.
If you are installing multiple instances of PingFederate on the same machine, such as a console node and an engine node in a clustered environment, install each instance using a unique `<pf_install>` directory.
If you are upgrading an existing PingFederate environment, see Upgrading PingFederate.

Click the corresponding tabs for your preferred installation method.

Installing PingFederate on Linux systems

Before you begin

- See System requirements for a list of qualified Linux operating systems.

- Request a license key through the Ping Identity licensing website.
- Ensure you are signed on to your system with sufficient privileges to install and run an application. You must install and run PingFederate under a local user account.
- Verify that you have installed the Java Runtime Environment (JRE) and that you have set the required environment variables correctly. For more information, see Installing Java.

About this task

To install PingFederate on a Linux system using the distribution `.zip` archive:

Steps

1. Download the latest version of the PingFederate Server distribution `.zip` archive from the Downloads website.
2. Extract the archive into the target installation directory.
3. Start PingFederate manually by running `<pf_install>/pingfederate/bin/run.sh`.

   > **Tip**
   >
   > To configure PingFederate to run as a service on Linux, install PingFederate on Linux manually.

   Result:

   The startup process is complete when you see the following message.

   ```
   PingFederate running...
   ```

Next steps

If your organization plans to manage keys and certificates using a hardware security module (HSM), see Supported hardware security modules.

Related links

- memoryoptions and installation
- Fine-tuning JVM options

Installing the PingFederate service on Linux manually

Before you begin

- Request a license key through the Ping Identity licensing website.
- Ensure you are signed on to your system with sufficient privileges to install and run an application.
- Verify that you have installed the Java Runtime Environment (JRE) and that you have set the required environment variables correctly. For more information, see Installing Java in the PingFederate Server documentation.

About this task

If you have not installed PingFederate on Linux using the distribution `.zip` archive, you can install it manually. To install the PingFederate service on Linux manually:

Steps

1. Download the distribution `.zip` archive from the Ping Identity website.
2. Extract the file into an installation directory, `<pf_install>`.
3. Create a new local user account for the PingFederate service, such as `pingfederate`.

   > **Note**
   >
   > The service account is referred to as *<pf_user>*.

4. Change the ownership of the PingFederate installation directory `<pf_install>` and update the read-write permissions by running the following commands:

   ```
   chown -R <pf_user>  <pf_install>
   chmod -R 775  <pf_install>
   ```

5. If the operating system supports systemd, install the PingFederate unit file:
   1. Edit the `<pf_install>/pingfederate/sbin/linux/pingfederate.service` systemd unit file.

      Replace the following variables with information from your environment:

      ${PF_VERSION}

      The version of PingFederate.

      ${PF_USER}

      The local user account for the PingFederate service.

      ${PF_HOME}

      The `<pf_install>/pingfederate` directory.For example, if `<pf_install>` is `/opt/identity.fed`, replace `${PF_HOME}` with `/opt/identity.fed/pingfederate`.

      ${PF_JAVA_HOME}

      The *<JAVA_HOME>* environment variable value (a directory).
   2. Copy the `pingfederate.service` file to the systemd unit files directory, for example, `/etc/systemd/system`.

      > **Note**
      >
      > Depending on the operating system, the exact location might vary. Consult your system administrators as needed. The rest of the step assumes `/etc/systemd/system` is the systemd unit files directory.

   3. Run the following command to update the read-write permissions of the `pingfederate.service` systemd unit file:

```
chmod 664 /etc/systemd/system/pingfederate.service
```

4. Run the following commands to load the new system configuration changes and start the PingFederate service:

```
systemctl daemon-reload ;\
systemctl start pingfederate
```

5. Run the following commands to configure the PingFederate service to start automatically as the server boots:

```
systemctl enable pingfederate ;\
systemctl daemon-reload ;\
systemctl restart pingfederate
```

Result:

After setting up the PingFederate systemd unit file, you can run the following **`systemctl`** command to manage the PingFederate service:

```
systemctl start pingfederate
systemctl stop pingfederate
systemctl restart pingfederate
systemctl status pingfederate
```

6. If the operating system supports SysV initialization, follow these steps to install the PingFederate script.

1. Edit the `<pf_install>/pingfederate/sbin/linux/pingfederate` script.

Replace the following statements with information from your environment:

PF_HOME=$PF_HOME

Replace *$PF_HOME* with the `<pf_install>/pingfederate` directory.For example, if `<pf_install>` is `/opt/identity.fed`, replace *$PF_HOME* with `/opt/identity.fed/pingfederate`.

USER="pingfederate"

If the PingFederate service account is not `pingfederate`, replace *<pingfederate>* with the local user account for the PingFederate service.For example, if *<pf_user>* is `pingfed`, replace *<pingfederate>* with `pingfed`.

Example:

Example (truncated)

If `<pf_install>` and *<pf_user>* are `/opt/identity.fed` and `pingfederate` respectively, the required modifications are:

```
...
PF_HOME=/opt/identity.fed/pingfederate
DIR="$PF_HOME/sbin"
USER="pingfederate"
...
```

2. Copy the `pingfederate` script to the SysV initialization directory, for example, `/etc/rc.d/init.d`.

   The exact location might vary, depending on the operating system. Consult your system administrators, as needed. The rest of the step assumes `/etc/rc.d/init.d` is the SysV initialization directory.

3. Run the following command to update the read-write permissions of the `pingfederate` SysV initialization script:

   ```
   chmod 755 /etc/rc.d/init.d/pingfederate
   ```

4. Configure the operating system to start the PingFederate service at various runlevels.

   On an RHEL server, you can use the **Service Configuration** utility to do so.

   Alternatively, the initialization directories associated with various runlevels can accept manual symbolic links of the `pingfederate` script by running the `ln -s <source> <target>` command.

   Example:

   You can create the following symbolic links on an RHEL server where runlevels 2 and 4 are not used:

   ```
   ln -s /etc/rc.d/init.d/pingfederate /etc/rc3.d/S84pingfederate
   ln -s /etc/rc.d/init.d/pingfederate /etc/rc5.d/S84pingfederate
   ln -s /etc/rc.d/init.d/pingfederate /etc/rc0.d/K15pingfederate
   ln -s /etc/rc.d/init.d/pingfederate /etc/rc1.d/K15pingfederate
   ln -s /etc/rc.d/init.d/pingfederate /etc/rc6.d/K15pingfederate
   ```

> **Note**
>
> Some operating systems might require a restart of the system to activate the new scripts. Consult your system administrators as needed.

Next steps

After setting up the PingFederate SysV initialization script, you can use the **`Service Configuration`** utility or run the following **`service`** commands to manage the PingFederate service:

```
service pingfederate start
service pingfederate stop
service pingfederate restart
service pingfederate status
```

Related links

- memoryoptions and installation

- Fine-tuning JVM options

Installing PingFederate on Windows

Before you begin

- Request a license key through the Ping Identity licensing page.
- Ensure you are signed on to your Windows system with sufficient privileges to install and run an application. PingFederate needs read, write, and execute permissions to function.
- Verify that you have installed the Java Runtime Environment (JRE) and that you have set the required environment variables correctly. Learn more in Installing Java.

About this task

You can install PingFederate on a Windows system using the installer for Windows or the distribution `.zip` archive. Using the installer for Windows is the preferred method.

> **Tip**
>
> To configure PingFederate to run as a service on Windows, install PingFederate on Windows manually. Learn more in Installing the PingFederate service on Windows manually.

To install PingFederate:

Steps

1. Install PingFederate using the installer for Windows or the distribution `.zip` archive:

   Choose from:

   - To install using the PingFederate installer for Windows:
     1. Download the PingFederate installer for Windows from the Ping Identity website.
     2. Double-click the `.msi` file to open the PingFederate **Setup Wizard**, and follow the instructions to complete the installation.

        PingFederate is configured to run as a service and starts automatically at the end of the installation process.

        > **Note**
        >
        > The PingFederate installer for Windows installs only one instance of PingFederate on a Windows server. If you need additional PingFederate instances on the same Windows server, install them using the distribution `.zip` archive.
        > You must manually configure various port settings in the `<pf_install>/pingfederate/bin/run.properties` file for each instance to avoid port conflicts.
   - To install PingFederate using the distribution `.zip` archive:
     1. Download the distribution `.zip` archive from the Ping Identity website. The distribution `.zip` archive is identical for both Windows and Linux.

    2. Extract the file into an installation directory.

2. If you have installed PingFederate by extracting the distribution `.zip` archive, start PingFederate manually by running `<pf_install>/pingfederate/bin/run.bat`.

   Wait for the script to finish. The startup process completes when you see the following message.

   ```
   PingFederate running...
   ```

   > **Caution**
   >
   > When running PingFederate for Windows, switching the Java version from 8 to 11 (or the reverse) prevents the service from running, and you won't be able to start PingFederate. The problem occurs because garbage collection logging configuration arguments that are used by Java 8 are incompatible with those used by Java 11. To change Java versions:
   >
   > 1. Run `<pf_install>\pingfederate\sbin\win-x86-64\uninstall-service.bat` to de-register the PingFederate service.
   > 2. Install the new Java version and update the *JAVA_HOME* and *PATH* environment variables.
   > 3. Run `<pf_install>\pingfederate\sbin\win-x86-64\install-service.bat` to register the PingFederate service.

Next steps

If your organization plans to manage keys and certificates using a hardware security module (HSM), see Supported hardware security modules.

Related links

- memoryoptions and installation
- Fine-tuning JVM options

Installing the PingFederate service on Windows manually

Before you begin

- Request a license key through the Ping Identity licensing page.
- Ensure you are signed on to your Windows system with sufficient privileges to install and run an application. PingFederate needs read, write, and execute permissions to function.
- Verify that you have installed the Java Runtime Environment (JRE) and that you have set the required environment variables correctly. Learn more in Installing Java.

About this task

> **Note**
>
> If you have installed PingFederate using the installer for Windows, skip these steps because PingFederate has already been configured to run as a service and to start automatically at the end of the installation process.

To install the PingFederate service manually:

Steps

1. Download the distribution `.zip` archive from the Ping Identity website.

   The distribution `.zip` archive is identical for both Windows and Linux.

2. Extract the archive into an installation directory, `<pf_install>`.

3. Start PowerShell or Command Prompt as an administrator.

4. Run the `<pf_install>\pingfederate\sbin\win-x86-64\install-service.bat` file.

5. Go to **Control Panel > Administrative Tools > Services** to open the management console.

6. Right-click the **PingFederate** service and select **Start**.

Result

The PingFederate service starts automatically on reboot.

Related links

- memoryoptions and installation
- Fine-tuning JVM options

Uninstalling PingFederate

Learn how to uninstall PingFederate from a Windows or Linux server.

Uninstalling PingFederate involves removing the previously-installed PingFederate service and the installation directory, `<pf_install>`.

Click the corresponding tabs for instructions on uninstalling from Windows or from Linux.

Uninstalling from Windows

Uninstalling PingFederate from a Windows server

Before you begin

- Ensure you are signed on to your system with sufficient privileges to uninstall an application.
- Optionally, make a backup copy of the PingFederate installation directory `<pf_install>`.

About this task

The method you use to uninstall PingFederate depends on whether you installed it using the installer for Windows or the distribution `.zip` file.

Steps

1. Verify the installation medium in **Control Panel > Uninstall a Program**.

   > **Note**
   >
   > The existence of a PingFederate entry indicates a previous installation of PingFederate using the Windows installer. If there is no existing PingFederate entry, the distribution `.zip` file was used to perform the installation.

2. Depending on the type of installation, uninstall PingFederate using one of the following methods:

   Choose from:

   - To uninstall PingFederate using the PingFederate installer for Windows, go to **Control Panel > Uninstall a Program**.

   Doing this removes the PingFederate service and the installation directory.

   - To uninstall PingFederate using the distribution `.zip` file:
     1. Go to **Control Panel > Administrative Tools > Services** to open the management console.
     2. Right-click the **PingFederate** service and select **Stop**.
     3. Run `uninstall-service.bat` from the `<pf_install>\pingfederate\sbin` subdirectory that corresponds to your platform processor.
     4. Optionally, remove the PingFederate installation directory `<pf_install>`.

Uninstalling from Linux

Uninstalling PingFederate from a Linux server

Before you begin

- Ensure you are signed on to your system with sufficient privileges to uninstall an application.

About this task

You can use the systemd service or the SysV initialization script to uninstall PingFederate from a Linux server.

Steps

1. Uninstall PingFederate using one of the following methods:

   Choose from:

   - To uninstall PingFederate using the systemd service, use the following **`systemctl`** commands to stop and disable PingFederate:

     ```
     systemctl stop pingfederate ;\
     systemctl disable pingfederate ;\
     systemctl daemon-reload
     ```

     You can also remove the PingFederate systemd unit file `pingfederate.service` from the systemd unit files directory `/etc/systemd/system` before running the **`systemctl daemon-reload`** command.

   - To uninstall PingFederate using the SysV initialization script:

     Do one of the following:

     - Use the **`Service Configuration utility`** to stop and disable the PingFederate service.
     - Remove any symbolic links from various initialization directories to stop the PingFederate service.
     - Remove the PingFederate SysV initialization script `pingfederate` from the SysV initialization directory `/etc/rc.d/init.d`.

   > **Note**
   >
   > Depending on the operating system, the exact directory locations might vary. Consult your system administrators as needed.

2. **Optional:** Remove the PingFederate installation directory, `<pf_install>`.

Upgrading PingFederate

Use this section to learn how to upgrade your PingFederate environment to the latest version on Windows and Linux systems.

Depending on your PingFederate installation, you can upgrade by using the PingFederate installer for Windows or the Upgrade Utility, which automatically migrates existing PingFederate installations of 6.0 and later to the latest version. The Upgrade Utility is included with the software distribution.

For general information about upgrading Ping Identity products and an upgrade planning guide, see Planning your upgrade.

> **Important**
>
> Before upgrading your production environment, you should:
>
> - Upgrade your test environment and verify that the new installation meets your expectations.
> - Thoroughly retest the behavior of any customized components.
>
> After you complete the upgrade process, you can create a backup of your previous installation and remove it from the server.
> End users might experience service disruptions as you upgrade your PingFederate environment. As needed, schedule a maintenance window to perform the upgrade.

Upgrade paths

Operating system	Source version	Source installation medium	Possible upgrade paths
Microsoft Windows	Step 8.x through 11.x	PingFederate installer for Windows	PingFederate installer for Windows or PingFederate Upgrade Utility
	Step 6.x through 11.x	PingFederate product distribution `.zip` archive	PingFederate Upgrade Utility
Linux	Step 6.x through 11.x	PingFederate product distribution `.zip` archive	PingFederate Upgrade Utility

If you are upgrading from PingFederate 5.3 or earlier, contact Support for more information.

Both the PingFederate installer for Windows and the Upgrade Utility create a new installation based on the new product distribution `.zip` archive and then copy the relevant files and property values from the existing installation (the source) to the new installation (the target). As a result, neither tool affects the source installation.

Integration kits

Both upgrade tools also copy the program files for the deployed adapters, connectors, and token translators (the integration kits in general) from the source installation to the target installation. Although the tools don’t upgrade the integration kits automatically, you can download newer versions from the Ping Identity Downloads website and upgrade the integration kits manually. For integration kit documentation, see the Ping Identity website.

Upgrade FAQ

Planning your upgrade

I want to upgrade PingFederate. What resources can help with this?

The following links contain helpful resources to plan your PingFederate upgrade:

> **Tip**
>
> Use the version selector on the upper left of the page to select the documentation for the target version of PingFederate.

- Release Notes for the target version and any versions between your current version and the target
- Upgrade guide for the target version
- Upgrade considerations for the target version and any major releases between your current version and the target
- Post-upgrade tasks
- Upgrading PingFederate in a DevOps environment, if you have PingFederate deployed in a DevOps environment
- Updating to the latest maintenance release if you're performing an in-place maintenance upgrade

I'm upgrading from a legacy version of PingFederate, how do I start?

If you're running PingFederate 8.4 or older, you should upgrade to version 8.4 first. After you've upgraded and tested your 8.4 deployment, you can upgrade to a later version.

> **Important**
>
> PingFederate versions 7.x and earlier can't upgrade directly to PingFederate 12.x. You must perform an incremental upgrade.

How can I upgrade PingFederate with the least amount of downtime?

PingFederate doesn't support a fully zero downtime upgrade, but you can perform a near-zero downtime upgrade.

Downloads

I want to upgrade to a specific release that's no longer available on the Ping Identity downloads page. How can I download the version I need?

Submit a support case to request the release you need, and we'll provide you with a link.

Do I need to download the Upgrade Utility separately?

All supported releases include the Upgrade Utility in the PingFederate package, so there's no need to download it separately.

> **Important**
>
> Releases prior to 10.x don't include the Upgrade Utility, so you must download that separately.
> For example, if you want to upgrade to 8.4, you need to download the PingFederate 8.4 package and the PingFederate 8.4 Upgrade Utility package.

Performing the upgrade

How do I perform a major version or minor version upgrade?

You can upgrade to the next major or minor version using the PingFederate Upgrade Utility or the Windows installer. The Upgrade Utility is generally the preferred method.

How do I perform a maintenance release upgrade?

If you're upgrading to a maintenance release for PingFederate 10.x or later, you can upgrade by deploying the in-place maintenance update package. You can also use the PingFederate Upgrade Utility or the Windows installer.

Can I use the Windows MSI Installer to upgrade?

Yes, but only if PingFederate was intially installed using this method. If it wasn't, or if you're unsure how PingFederate was installed, use Upgrade Utility.

Do I have to use the Upgrade Utility to upgrade to major or minor releases in a DevOps environment?

Yes. In a DevOps environment, you must always use the Upgrade Utility. Learn more in the DevOps upgrade guide.

Can I use a configuration archive or bulk import from a previous release into a newer release?

PingFederate 12.1 and earlier: No. The Upgrade Utility or Windows installer are the only supported upgrade methods for major or minor releases. The Upgrade Utility is generally the preferred method.

PingFederate 12.2 and later: A configuration archive from version 11.1 or later will be upgraded automatically when imported. Importing an archive from a version earlier than 11.1 won't be upgraded or supported.

> **Note**
>
> This method only imports data stored in the configuration archive, which excludes things like templates, `.properties`, or `.conf` file settings, and other settings that the Upgrade Utility would migrate. As such, we recommend this method only for environments with appropriate automation to merge these configurations post-upgrade.

Do I need to start my upgrade with the Admin Console?

Yes. The Admin Console is the only interface that inherits the configuration during the upgrade (such as adapter, connections, and datastores). Failing to upgrade and start the Admin Console and perform cluster replication results in upgraded engine nodes missing key configuration changes needed to process transactions.

Licensing

Do I need to upgrade my license when performing an upgrade?

Yes, if you're upgrading to a new major release. You can upgrade your license by following this guide.

Does upgrading my license invalidate my existing license?

No. Your existing license continues to be valid until the expiration date.

General

How do I know if the version I am running or upgrading to is supported?

You can check the support status and end-of-life dates for PingFederate at the End of Life tracker.

Learn more in Ping Identity's End of Life Policy.

Does PingFederate support a mixed-version cluster?

You can mix maintenance releases on a cluster but not major and minor releases.

- PingFederate servers running 12.1.2 and 12.1.4 can exist within the same cluster. This makes it easier to upgrade to a new maintenance release. We recommend upgrading your older versions rather than keeping the mixed-version cluster long term.
- PingFederate servers running 12.1 and 12.2 won't communicate with each other.
- PingFederate releases prior to 10.x don't support any form of mixed-version clustering.

Do I need to update Java when I upgrade PingFederate?

Possibly. Check the system requirements for the target version to see whether your current JDK release is supported.

Does a PingFederate upgrade also upgrade my Adapters and Integration Kits?

By default, the Upgrade Utility migrates the existing Integration Kit versions from the source version. This is to maintain functionality. You can then upgrade to new Integration Kit versions after your PingFederate upgrade.

You can use the `-c` custom mode when running the Upgrade Utility. This prompts the administrator to choose to use the existing or new version of the integration, assuming the target PingFederate package includes the newer version of that integration.

Integration kits not included in the PingFederate package are always migrated from the source version.

Will my upgrade affect my existing use cases and functionality?

Possibly. Review the Upgrade considerations before upgrading, and perform the upgrade in a test environment so you can test your use cases before upgrading a production environment.

What happens to state and Authentication Sessions when upgrading?

When performing a major or minor version upgrade, state and sessions held in memory will be lost. Persistent Sessions are stored in an external database and will be available post-upgrade.

Issues and Rolling Back

After running the Upgrade Utility, when I start the Windows Service, the old PingFederate version starts up. Why?

The Upgrade Utility doesn't upgrade the Windows Service.

- You can uninstall the existing Windows Service and install the Windows Service for the new version.
- Alternatively, you can install the new Windows Service with a unique name so that both services are available to start and stop as you need.

How can I roll back a PingFederate upgrade?

PingFederate upgrades using the Upgrade Utility are non-destructive, meaning the previous version remains intact alongside the newly upgraded release. As such, you can roll back the upgrade by stopping the new service and starting the old service.

If I roll back my upgrade, do I have to rerun the Upgrade Utility when we are ready to retry the upgrade?

If your environment has changed since the last upgrade, you should run the Upgrade Utility again to ensure these chages are carried over.

How can I get help if I encounter issues with the new version post-upgrade?

Submit a support case with a description of the problem, along with the `server.log` and `upgrade.log` files and any other relevant details.

If this is a production environment and the issue is causing a significant impact, you should roll back the upgrade to restore functionality while the case is investigated.

I attempted to upgrade using an unsupported method but encountered problems. What can I do?

If you attempted to upgrade using an unsupported method, like importing a configuration archive from an older version into a new version or from an incompatible version into 12.2 or later, you should start by rolling back to the previous version and upgrade using the Upgrade Utility.

Upgrading using unsupported methods frequently causes problems, which can be difficult to diagnose.

Ping Support can't assist with problems caused by an unsupported upgrade process.

After upgrading, I get a warning banner in the PingFederate Admin Console regarding the HyperSQL Database (HSQLDB). What's causing this?

This banner is expected if you use the unsupported HSQLDB in your environment. Learn more in Hypersonic database Usage with PingFederate.

Preparing to upgrade PingFederate

Prepare to upgrade PingFederate by completing a number of tasks.

Steps

- Review the PingFederate release notes for enhancements, upgrade considerations, deprecated features, and other known issues and limitations.
- Review the post-upgrade tasks.
- Review potential changes in system and port requirements.
- Obtain a new license key if needed.
- Update the Java runtime environment (JRE) to version 8 or 11 on your PingFederate servers if needed.

  When you upgrade the JRE, modify the previously defined paths for the system *<JAVA_HOME>* and *<PATH>* environment variables.

  > **Important**
  >
  > PingFederate 7.2 and earlier will not start using the currently supported Java runtime. If you need to start the previous PingFederate version on the same server after the upgrade, retain the older Java installation and change environment variables back when needed.

- Complete any unfinished connections (Drafts) in the administrative console if you want to include them in the migration.

Upgrade considerations

The following modifications since PingFederate 11.0 might affect existing deployments.

SAML SP connection configuration

Existing SAML SP connections that rely on multiple session states in a single transaction will be affected by new session state validation measures introduced in PingFederate 11.2.5 and 11.3 under PF-33168. For more information, see PingFederate 11.3 (June 2023).

For more information about how to diagnose and resolve issues caused by this update, see Solicited SAML Response Validation in the Ping Identity Support Portal.

New token kid for RSA256 signing algorithm

With the added support of the `alg` attribute at the JWKS endpoint in PingFederate 11.3, the RSA signing key will now be listed an additional time, with the `_RS256` suffix and the `alg:RS256` attribute.

While signing ID tokens or access tokens using centralized signing keys, this new kid will be used when RSA256 is the preferred algorithm.

11.3.1 upgrade consideration

When updating to the latest maintenance release using an in-place update method (for example, from 11.3.0 to 11.3.x), in addition to the steps in Updating to the latest maintenance release, you must remove the existing PingFederate Windows service. After removal, re-install the PingFederate Windows Service to apply this fix.

Apache Velocity Engine upgrade

With PingFederate 11.3, we've upgraded the Apache Velocity templating engine from version 1.6 to version 2.3. All out-of-the-box PingFederate HTML templates are backward compatible with this new version of Velocity, but customized templates might require edits to work with version 2.3. For a complete list of changes to the Velocity engine that might require updating customized templates, see Upgrading in the Apache Velocity Engine documentation.

Replacement of `hivemodule.xml`

Starting with PingFederate 11.3, the distribution ZIP file no longer contains the `hivemodule.xml` file. Instead, the distribution contains a new file: `service-points.conf` in the `<pf_install>/pingfederate/server/default/conf/` directory. On first startup, PingFederate processes `service-points.conf` and generates a default `hivemodule.xml`.

The advantage of the `service-points.conf` file is that it lets you use environment variables to override hivemodule settings without having to modify an XML file. For more information, see Overriding configuration settings using environment variables.

When you upgrade a supported version of PingFederate older than 11.3 to version 11.3 or later, the `service-points.conf` file replaces the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF/` directory with a generated default `hivemodule.xml` file in `<pf_install>/pingfederate/server/default/conf/generated-hivemodule/META-INF/` directory.

If you want to continue using `hivemodule.xml` instead of `service-points.conf`, add `hivemodule.xml` to the directory `<pf_install>/pingfederate/server/default/conf/META-INF/` after upgrading PingFederate. During startup, if PingFederate finds `hivemodule.xml` in that directory, it will ignore `service-points.conf`. Later, when you upgrade PingFederate 11.3 or a newer version, the `hivemodule.xml` file won't be replaced.

These changes don't affect your upgrade process in either of the following cases:

- You upgrade to PingFederate 11.3 or later using the upgrade utility, as recommended
- You don't need to modify the default `hivemodule.xml` file

> **Important**
>
> If you start a new instance of PingFederate 11.3 or later using a process that depends on having `hivemodule.xml` in the distribution ZIP file, the process will fail. To prevent this startup failure, do one of the following:
>
> - Launch a local instance of the new PingFederate version, which generates the default `hivemodule.xml` in `<pf_install>/pingfederate/server/default/conf/generated-hivemodule/META-INF/`. Copy and, if needed, modify `hivemodule.xml`. Use that file to launch your production PingFederate instance. When you launch PingFederate, your existing process can grab `hivemodule.xml` and work as expected.
> - Disable your process that needs to work with `hivemodule.xml` directly. If needed, modify the `service-points.conf` file that comes with PingFederate 11.3 or later. When you launch PingFederate, it generates `hivemodule.xml` and works as expected.
> - Disable your process that needs to work with `hivemodule.xml` directly. If needed, configure the PingFederate environment variables. When you launch PingFederate, it uses those environment variables and generates `hivemodule.xml`.

LDAP properties

Starting with PingFederate 11.3, the `<pf_install>/pingfederate/bin/ldap.properties` file includes the new `ldap.type` field. This mandatory field specifies the LDAP directory server type. If you use LDAP for administrative console or administrative API authentication, ensure you set this field to a valid value.

Refresh token rolling grace period

When upgrading from PingFederate versions later than 11.0.1, if the **Authorization Server Settings** window was never modified, a default value of 0 shall be set for **Refresh Token Rolling Grace Period**. If the **Authorization Server Settings** window was modified, the previously set value shall be retained. For more information, see the Refresh Token Rolling Grace period field in Configuring authorization server settings.

Metadata response configuration

Starting with PingFederate 11.2, PingFederate provides an OAuth authorization server metadata endpoint, `/.well-known/oauth-authorization-server`. The new endpoint returns configuration information that OAuth clients need to interface with PingFederate using the OAuth 2.0 protocol. You can customize what information the endpoint returns by editing the `<pf_install>/pingfederate/server/default/conf/openid-configuration.template.json` template file, which also determines what information the OpenID Connect metadata endpoint returns.

If you previously modified `openid-configuration.template.json` for the OpenID Connect metadata endpoint, merge your modifications into the 11.2 version of the template.

Processing policy fragments

Starting with PingFederate 11.2, PingFederate processes policy fragments independently from policies and other fragments. The nodes in a fragment are no longer aware of the exterior policy or fragment invoking it, other than the fragment input values that the input authentication policy contract provides. The inverse is also true. The exterior policy or fragment is no longer aware of the processing that occurred within an interior fragment, only what is mapped to the output authentication policy contract.

Integration with Amazon Web Services (AWS) CloudHSM

Starting with PingFederate 11.2, PingFederate's integration with AWS CloudHSM now requires the AWS CloudHSM Java Cryptography Extension (JCE) provider for Client SDK 5. There are a number of new limitations around assertion and token decryption with Client SDK 5. For more information on limitations, see the Hardware security modules (HSM) issue section of the Release Notes. In addition, the `java.security` file in the Java Runtime Environment (JRE) no longer needs to be modified and client files do not need to be copied to the `JAVA_HOME/jre/lib/ext` directory. For more information on how to upgrade the HSM client, see the updated setup instructions in Integrating with AWS CloudHSM.

Verbose logging

Starting with PingFederate 11.2, you can enable verbose logging with the **Log Settings** window or with the administrative API on nodes that have the updated version of the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file. If `log4j2.xml` was previously modified, PingFederate will migrate the modified file on upgrade and create a backup of the latest file. To take advantage of the new verbose logging capability, you must merge your modifications into the latest version of `log4j2.xml`.

Expired passwords and prompts to change passwords

When using PingDirectory 8.2 or later as the credential store, if a user's password has expired, the following settings ensure PingFederate doesn't direct the user to the Change Password form when they enter an invalid password:

- Set the **return-password-expiration-controls** setting in the PingDirectory password policy to **always**.
- Starting with PingFederate 11.1, when configuring an LDAP Username Password Credential Validator, select the **Expect Password Expired Control** checkbox in the advanced fields section of the **Instance Configuration** tab of the **Create Credential Validator Instance** window.

Device authorization flows using the authentication API

Starting with PingFederate 11.1, the user code verification steps of the device authorization flow are API-enabled. Existing API applications will need to be updated to support the new API states. To disable the use of the API for the new states, set `enable-authn-api-for-user-code-validation` to `false` in `<pf_install>/pingfederate/server/default/config-store/oauth-device-flow.xml`. This will cause PingFederate to continue to render these states itself using the Velocity templates.

Java 17

Starting with PingFederate 11.1, when using Java 17, the following JDKs are supported:

- Adoptium OpenJDK
- Oracle JDK
- Amazon Correto

If you upgrade to Java version 17 after you upgrade PingFederate:

- Remove `-XX:-UseParallelOldGC` from the `jvm-memory.options` file.
- Reinstall the Windows Service.

Certificate revocation checking

Starting with PingFederate 11.1, when configured for certificate revocation list (CRL) or online certificate status protocol (OCSP) revocation checking, PingFederate now performs these checks in a broader range of flows. In particular, outbound TLS calls to retrieve JSON web key sets (JWKS) now perform revocation checks. Also, outbound TLS calls performed by plugins, such as adapters, now perform revocation checks. To revert to the previous behavior where PingFederate doesn't perform these new checks, set `pf.tls.installRevocationCheckerGlobally` to `false` in the `run.properties` file.

> **Note**
>
> CRL-based revocation checks can consume large amounts of memory. If CRL checking is enabled, you should set PingFederate's maximum heap in the `jvm-memory.options` file to at least 4 GB. Alternatively, consider switching to OCSP revocation checking, which demands fewer resources.

When configured for CRL or OCSP revocation checking, PingFederate's validation of CRLs and OCSP responses is tighter than before. The following settings in the `revocation-checking-config.xml` file control the new validation checks:

- `crl-issuer-allow-any-trust-anchor`
- `crl-verify-issuing-distribution-point`
- `OCSP-enforce-responder-key-usage-check`

Although you should leave the new validation checks enabled, you can disable them for compatibility with existing deployments.

Custom `MasterKeyEncryptor` *implementations*

Starting with PingFederate 11.1, due to the revocation checking enhancements outlined above, a circular initialization dependency can arise if revocation checking is enabled and a custom `MasterKeyEncryptor` implementation makes API calls to an external service. To avoid this risk, you should disable revocation checking for any API calls made by the master key encryptor. Refer to the SDK documentation for more information on how to disable revocation checking in custom `MasterKeyEncryptor` implementations.

`run.sh` *and* `run.bat` *files*

In PingFederate 11.1, the `run.sh` and `run.bat` files were updated.

- A `startup` directory was added under the PingFederate installation. Its contents are now added to the classpath using a wildcard.
- References to `jetty-start.jar` were updated in `run.sh` and `run.bat` because `jetty-start.jar` was moved from the `bin` directory to the `startup` directory.
- References to `run.jar` were removed from `run.sh` and `run.bat` because `run.jar` was removed from the `bin` directory.

  If your `run.sh` or `run.bat` files were customized, you must update your copy of these files accordingly.

Template `username.recovery.template.html`

Starting with PingFederate 11.1, the `username.recovery.template.html` template no longer includes the *$forgotPasswordUrl* variable.

> **Tip**
>
> The top of the template file documents which variables you can use.

Dynamic discovery settings

In versions preceding PingFederate 11.0, administrators could only define dynamic discovery settings to discover cluster membership in the `server/default/conf/tcp.xml` file. Now PingFederate provides a new configuration file for these settings, `bin/jgroups.properties`. This new approach streamlines future upgrade experiences. For new installations, we recommend defining dynamic discovery settings in the `jgroups.properties` file. While upgraded environments will continue to look for dynamic discovery settings from the `tcp.xml` file, we recommend performing a one-time migration to ease the upgrade experiences in the future. For more information, see Migrating cluster discovery settings.

Velocity HTML templates

Starting with PingFederate 11.0, if any of the default Velocity HTML templates for user-facing windows were modified, the Upgrade Utility migrates them to the new installation and renames the corresponding default templates in the new installation with the following format: `<template_name>-default-<PF-version>.<ext>`. For more information, see User-facing windows.

Kerberos authentication

Starting with PingFederate 11.0, when the new **Retain Previous Keys on Password Change** check box on the **Manage Domain/Realm** window is selected, PingFederate saves the encryption keys associated with the password of the current Kerberos service account. The check box is selected by default.

> **Note**
>
> PingFederate will not save the encryption keys until you re-save the configuration of the domain or realm. To facilitate seamless rotation of the service account password for existing domains, click **Save** on the **Manage Domain/Realm** window before you update the password in the domain controller. For more information, see Adding Active Directory domains and Kerberos realms.

IWA IdP adapter

PingFederate 11.0 and later no longer support the integrated Windows authentication (IWA) IdP adapter. The IWA integration kit for Kerberos has been replaced with a PingFederate adapter for Kerberos. See Migrating from the Integrated Windows Authentication Integration Kit to the PingFederate Kerberos adapter.

Private key JSON web token authentication

When authenticating an OAuth client that uses the private key JSON web token (JWT) authentication scheme, PingFederate now validates that the issuer and subject claims in the JWT have the same value.The following administrative API endpoint exposes the validation on/off switch:

```
https://{{pf_base_host_port}}/pf-admin-api/v1/configStore/oauth-credentials-validator/
issuerMustBeEqualToClientId
```

To disable validation, send an HTTP POST request with the following body to the endpoint:

```
 {
   "id": "issuerMustBeEqualToClientId",
   "stringValue": "false",
   "type":"STRING"
 }
```

Authentication API applications

Starting with PingFederate 11.0, the new **Restrict Access to Redirectless Mode** check box on the **Authentication API Applications** window lets you restrict which authentication API applications can use redirectless mode. To avoid impacting existing deployments, this check box is not selected on upgrade. However, you should enable this setting. For more information, see Managing authentication applications.

Jetty agent

Starting with PingFederate 11.0, if your PingFederate server is running on a Java version prior to 8u252, you must modify your `run.sh`, `run.bat`, or `PingFederateService.conf` script to include the new Jetty agent in PingFederate.

Add the following Java argument to the script:

```
-javaagent:/server/default/lib/jetty-alpn-agent.jar
```

Example for `run.sh`:

```
"$JAVA" $JAVA_OPTS \
$ERROR_FILE \
$HEAP_DUMP \
${GC_FILE:+$GC_FLAG"$GC_FILE"$GC_OPTIONS} \
$ENDORSED_DIRS_FLAG \
 -javaagent:$PF_HOME/server/default/lib/jetty-alpn-agent.jar  \
-Dlog4j2.AsyncQueueFullPolicy=Discard \
```

Example for `run.bat`:

```
"%JAVA%" %PF_JAVA_OPTS% %JAVA_OPTS% %GC_OPTION%  -javaagent:%PF_HOME%/server/default/lib/
jetty-alpn-agent.jar  -Dlog4j2.AsyncQueueFullPolicy=Discard
```

Example for `PingFederateService.conf` (note the extra `../` because this is located in `pingfederate/sbin/wrapper`):

```
# Java Additional Parameters
wrapper.java.additional.1=-Dlog4j.configurationFile=../../server/default/conf/log4j2.xml

...... (omitted lines 2-13 to save space) ......

wrapper.java.additional.14=-javaagent:../../server/default/lib/jetty-alpn-agent.jar
```

Specifying a maximum size for inbound runtime requests

Starting with PingFederate 11.0, if you have previously specified a value for `maxFormContextSize` in `jetty-runtime.xml`, you should now use `pf.runtime.http.maxRequestSize` in the `run.properties` file to control the maximum size for inbound runtime requests. For more information, see Configuring PingFederate properties.

Java 8

As we continue to improve our products and hardware security module (HSM) integrations, you should migrate off of Java 8. We intend to remove Java 8 support from our qualification process in May 2023. For more information, including Java 11 support, see System requirements.

Third-party integrations

As we continue to improve PingFederate, we intend to remove the following product releases from our qualification process after the release of PingFederate 11.3 in June 2023:

- Oracle Linux 7.9 (Red Hat-compatible Kernel)
- Red Hat Enterprise Linux 7.9
- Microsoft SQL Server 2016 SP2
- Oracle Database 12c Release 2
- Microsoft Windows Server 2012 R2
- Microsoft Active Directory 2012 R2

We encourage you to upgrade these products to more recent versions, such as:

- Oracle Linux 8.5 (Red Hat-compatible Kernel)
- Red Hat Enterprise Linux 8.5
- Microsoft SQL Server 2017
- Oracle Database 19c
- Microsoft Windows Server 2016
- Microsoft Active Directory 2016

For a more complete list of qualified third-party solutions, see System requirements.

Upgrade considerations introduced in PingFederate 10.x

Several specific modifications since PingFederate 10.0 might affect existing deployments.

Delayed heartbeat response due to archive import on startup

Starting with version 10.2, when you place an archive in the `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory on startup, the heartbeat endpoint will not return `200` until archive import completes. Depending on how long archive import and configuration loading takes, the first successful heartbeat response may be significantly delayed relative to earlier versions. If you have configured a health check or probe that can trigger a restart of the server, crash loop behavior can result. Review the configuration of these checks to ensure time thresholds are set appropriately.

TLS 1.0 and 1.1 disabled

Starting with version 10.3, PingFederate disables TLS 1.0 and 1.1 for both inbound and outbound connections by default. As a result, clients using TLS 1.0 or 1.1 will no longer be able to connect to the administrative port or the runtime port. If you must re-enable TLS 1.0 or 1.1, add `TLSv1` or `TLSv1.1` to the `run.properties` file: look for the “TLS Protocol Settings” section and follow the inline instructions. Additionally, you might need to add back the weaker cipher suites, such as TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, or TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA. For more information, see Managing cipher suites.

Bouncy Castle FIPS mode

When upgrading an installation where Bouncy Castle FIPS mode is enabled, it is no longer necessary to place the `bc-fips` jar file in the `JAVA_HOME/jre/lib/ext` directory. It is also no longer necessary to modify the `JAVA_HOME/jre/lib/security/java.security` file. It is recommended to revert these changes to the Java environment.

SameSite cookie configuration

As of PingFederate 10.3, the Jetty configuration uses the native servlet SameSite cookie configuration. This moves the SameSite specifier declaration to its own attribute in the Jetty configuration as follows:

- New format for `jetty-admin.xml` in the DeploymentManager:

```
<Call name="setContextAttribute">
    <Arg>org.eclipse.jetty.cookie.sameSiteDefault</Arg>
    <Arg>None</Arg>
</Call>
```

- New format for `jetty-runtime.xml` in the WebAppContext:

```
<Call name="setAttribute">
    <Arg>org.eclipse.jetty.cookie.sameSiteDefault</Arg>
    <Arg>None</Arg>
</Call>
```

- If you want to specify a default value for session management cookies, such as JSESSIONID, in servlets hosted in PingFederate, add a `<comment>` like the one in the following snippet to the existing `session-config` in the `web.xml` file:

```
<session-config>
    <session-timeout>30</session-timeout>
        <cookie-config>
            <http-only>true</http-only>
            <!--
                The following comment adds a default SameSite value to the JSESSIONID cookie
in any servlet context.
                Available options are:
                    SAME_SITE_NONE
                    SAME_SITE_LAX
                    SAME_SITE_STRICT
            -->
         <comment>SAME_SITE_NONE</comment>
    </cookie-config>
</session-config>
```

Microsoft Internet Explorer 11

Ping Identity commits to deliver the best experience for administrators and users. As we continue to improve our products, we encourage our customers to migrate off of Microsoft Internet Explorer 11. We intend to remove Internet Explorer 11 from our qualification process in December 2021.

Microsoft Windows Server 2012 R2 and Active Directory 2012

Because Microsoft will end extended support for Windows Server 2012 R2 in late 2023 (see here), you should upgrade your Windows servers and Active Directory to a later version, such as Windows Server 2019. For a full list, see System requirements. We intend to remove Windows Server 2012 R2 and Active Directory 2012 from our qualification process in July 2023.

Authorization endpoint

Before version 10.2, PingFederate did not validate the `NumericDate` value of `exp` claims in a signed request object's JWT. To ensure the JWT does not expire too far in the future, PingFederate 10.2 and later do validate the value. PingFederate rejects any JWT that expires more than 720 minutes later. You can change that default value in `<pf_install>/pingfederate/server/default/data/config-store/jwt-request-object-options.xml`.

> **Note**
>
> PingFederate interprets the `NumericDate` value as seconds, not milliseconds. So PingFederate 10.2 will reject a JWT that has the `NumericDate` value based on milliseconds, because PingFederate calculates the JWT to live more than 720 minutes.

Configuration change necessary for MFA adapters

As of PingFederate 10.2, when you define policies using multi-factor authentication (MFA) adapters, you must select the **User ID Authenticated** check box in the **Incoming User ID** popup to allow users to register as a new MFA user. You should only select this check box if the previous authentication source has verified the **Incoming User ID**. You should not select the check box if the MFA adapter is part of a policy used for password reset or password change. For more information, see Defining authentication policies.

> **Important**
>
> Administrators using the PingID adapter must review existing policies and select this check box if appropriate. Otherwise, the adapter will prevent new user registration.

Expression Admin role

When upgrading to PingFederate 10.1 or later from an earlier version, administrative users who were granted the Admin role in the earlier installation are granted the Expression Admin role automatically. You can achieve the same result by using the `/bulk/import` administrative API endpoint to bulk-import a configuration that was bulk-exported from PingFederate 10.0.Additionally, all four administrative roles, namely User Admin, Admin, Expression Admin, and Crypto Admin, are required to access and make changes through the following services:

- The `/bulk`, `/configArchive`, and `/configStore` administrative API endpoints
- The **System** > **Server** > **Configuration Archive** window in the administrative console
- The **Connection Management** configuration item on the **Security** > **System Integration** > **Service Authentication** window

Authentication session created after user registration

As of PingFederate 10.1, an authentication session is automatically created for a user after registration, preventing the user from having to log in again during the next SSO transaction. This feature is enabled by default for all new and existing local identity profiles. However, if needed, you can disable it through the `/localIdentity/identityProfiles` administrative API endpoint by setting the `createAuthnSessionAfterRegistration` attribute to `false`.

Template `html.form.login.template.html`

Starting with PingFederate 10.0, the `html.form.login.template.html` template no longer includes the *$forgotPasswordUrl* variable.

Related links

- Upgrade considerations introduced in PingFederate 9.x
- Upgrade considerations introduced in PingFederate 8.x
- Upgrade considerations introduced in PingFederate 7.x
- Upgrade considerations introduced in PingFederate 6.x

Upgrade considerations introduced in PingFederate 9.x

Gemalto SafeNet Luna HSM 6.3

When integrating with Gemalto SafeNet Luna Network HSM 6 (hardware security module), PingFederate 9.2 requires firmware version of 6.3.0 and client driver version of 6.3. See Integrating with Thales Luna Network HSM for setup information.

Weaker cipher suites disabled

Starting with PingFederate 9.1, weaker cipher suites TLS_RSA_WITH_AES_128_CBC_SHA and TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA are disabled in new installations and upgrades. As a result, the administrative and runtime servers support only TLS 1.2. If you must re-enable these cipher suites for legacy clients, refer to Managing cipher suites for more information.

LDAP service accounts on PingDirectory

If PingFederate 9.3.1 or newer has an LDAP connection with PingDirectory, then add the config-read privilege to its service account in PingDirectory. Otherwise, users will not receive password expiry notifications. For more information, see Assigning Privileges to Normal Users and Individual Root Users in the PingDirectory documentation.

Improved validation for `AudienceRestriction`

If an IdP connection is configured with multiple virtual server IDs, the `AudienceRestriction` value in a SAML response must now match the virtual server ID information embedded in the protocol endpoint at which PingFederate receives the message. Otherwise the SSO attempt fails. To override this validation on a per-connection basis, see Configuring validation for the AudienceRestriction element.

Custom authentication selector

If you have created a custom authentication selector that returns an IdP adapter instance ID or the connection ID of an IdP connection, you must update the associated descriptor instance. Learn more in Updating the custom authentication selector.

Provisioning datastore reset

Upgrading to PingFederate 9.0 or 9.0.1 when using its outbound provisioning capability can result in user records being disabled at SaaS applications. The issue is resolved in version 9.0.2.

If you are upgrading from version 8.4.4 (or earlier) or from version 9.0.2, 9.0.3, and 9.0.4 to version 10.0, the upgrade process automatically resolves this issue. No further action is required.

If you are upgrading from version 9.0 or 9.0.1 to PingFederate 10.0, you must use the `provmgr` command-line tool to reset the provisioning datastore on the upgraded installation. See Reviewing database changes for more information.

Security enhancement in JDBC datastore queries

A security enhancement was made in PingFederate 9.0 to safeguard JDBC datastore queries against back-end SQL injection attacks. This protection is enabled for all new installations. For upgrades, see Reviewing database changes.

Access token validation response

Starting with PingFederate 9.2, the access token validation response no longer includes the username and subject elements by default. Responses include them only if they were mapped in the issuing access token management instance.

Upgrade considerations introduced in PingFederate 8.x

`InterReqStateMgmtMapImpl.expiry.mins` *renamed*

The `InterReqStateMgmtMapImpl.expiry.mins` setting in the `size-limits.conf` file has been renamed in PingFederate 8.4.2. If you have previously modified the value of this setting, please refer to Copying customized files or settings for more information.

An improved index (`IDX_FIELD_NAME`*) in the database table for OAuth clients*

PingFederate 8.4 has modified an existing index (`IDX_FIELD_NAME`) in the `pingfederate_oauth_clients_ext` database table as a general improvement. For information about modifying this index in your existing table, see Reviewing database changes.

Security enhancement to the OAuth token endpoint

Starting with version 8.3, a new PingFederate installation no longer allows OAuth clients to send access token validation requests to its token endpoint (`/as/token.oauth2`) by the HTTP GET method.For upgrades, the Upgrade Utility applies this new behavior to the new installation as well unless the `<pf_install>/pingfederate/server/default/data/config-store/oauth-token-endpoint-binding.xml` file has been modified in the older version, in which case the Upgrade Utility preserves the modified configuration.

SSLv2Hello disabled

Starting with PingFederate 8.3, SSLv2Hello is disabled. Customers are encouraged to update their applications to use TLSv1, TLSv1.1, or TLSv1.2 when establishing HTTPS connections with PingFederate.As needed, SSLv2Hello can be re-enabled as needed. See Enabling SSLv2Hello for more information.

License management simplification

Starting with version 8.2, PingFederate no longer maintains its license information in the `<pf_install>/pingfederate/server/default/data/.pingfederate.lic` file, which is known as the secondary license file in the previous versions of PingFederate. The `.pingfederate.lic` , if any, is ignored.

We recommend using the administrative console to simplify the license management aspect of a standalone PingFederate server or a clustered PingFederate environment.

Security enhancement for a clustered PingFederate environment

As of PingFederate 8.1, when encryption is enabled for the network traffic sent between nodes in a clustered PingFederate environment, you must provide an authentication password for the cluster as well; otherwise PingFederate aborts during its startup process.For more information about the `pf.cluster.encrypt` and `pf.cluster.auth.pwd` properties, see Deploying cluster servers.

Metadata signing

Previously, when no signing certificate was chosen on the **Metadata Signing** tab on the **System > Protocol Metadata > Metadata Settings** window, the `/pf/sts_mex.ping` and `/pf/federation_metadata.ping` system-services endpoints provided signed WS-Trust and WS-Federation metadata using one of the certificates configured on the **Security > Certificate & Key Management > Signing & Decryption Keys & Certificates** window.Starting with PingFederate 8.1, if no certificate is selected in the **Metadata Signing** menu, PingFederate provides unsigned metadata at both aforementioned endpoints. Select a certificate in the **Metadata Signing** window if signed metadata is desired.

Hostname verification for email server

For email notification using SSL or TLS, hostname verification of the certificate is available starting with PingFederate 8.1. This option is enabled automatically when the **Use SSL** or **Use TLS** check box is selected for a new configuration. When upgrading from a previous version of PingFederate, if email notification had already been configured to use SSL or TLS, the Upgrade Utility preserves the configuration without activating the hostname verification option for compatibility reasons. Administrators should consider activating this new option to improve security.

New login template file for the HTML Form Adapter

Previously, when multiple instances of the HTML Form Adapter are chained together (for example, in an instance of the Composite Adapter), the subsequent instance tried authenticating the end user with the credentials from the previous login, which might fail when the HTML Form Adapter instances were configured to use different password credential validators (PCVs). Although this use case is rare, PingFederate 8.1 has corrected the behavior. As a result, the login template file, `<pf_install>/pingfederate/server/default/conf/template/html.form.login.template.html`, has been modified.If you have previously customized this login template file and if you have authentication use cases that chain multiple instances of the HTML Form Adapter, you should re-customize using the new `html.form.login.template.html` file.

New connection pool library

As of PingFederate 8.0, support for BoneCP as the JDBC connection pool library has been deprecated and replaced with Apache Commons DBCP 2, which requires JDBC 4.1 or later drivers.Verify the database-driver JAR files, found in the `<pf_install>/pingfederate/server/default/lib` directory, meet the minimum version requirement. If you are using JDBC drivers of version 4.0 (or earlier), contact your vendors for the latest drivers and replace the older JDBC database-driver JAR files with the latest.For more information, including re-enabling BoneCP as the JDBC connection pool library, see Reviewing database changes.

Log4j 2 upgrade

PingFederate 8.0 has upgraded its logging framework from Log4j to Log4j 2.If you have previously customized `<pf_install>/pingfederate/server/default/conf/log4j.xml`, you will need to manually migrate your changes to the new `log4j2.xml` in the same `conf` directory. See Logging configurations for instructions.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers.

Upgrade considerations introduced in PingFederate 7.x

Hostname verification for LDAPS

For LDAP type datastores with LDAPS enabled, hostname verification of the certificate is enabled by default for all new datastores beginning with PingFederate 7.3. When upgrading from a previous version of PingFederate, this option is disabled for existing datastores for compatibility reasons. Administrators should consider activating this new option for greater security.

Changes in a database table supporting nested group membership

Outbound provisioning of groups and nested group membership requires an update in the internal datastore. Follow the instructions in Reviewing database changes to add or update the **group_membership** table.

SSLv3 disabled

To mitigate the POODLE attack, the SSLv3 protocol is disabled by default starting in PingFederate 7.3. It can be re-enabled by modifying the connector configuration in `jetty-runtime.xml` and `jetty-admin.xml` found in the `<pf_install>/pingfederate/etc` directory.

New representation for multivalued attributes in WS-Federation assertions

Starting with PingFederate 7.3, multivalued attributes in WS-Federation assertions are now represented as multiple `AttributeValue` elements under a single `Attribute` element. Previously, they were represented as a series of `Attribute` elements with the same name. The new behavior was implemented for compatibility with ADFS 2.0. To revert to the previous behavior, a setting is available in `wstrust-global-settings.xml`.

A new index (`EXPIRESIDX`) in the database table for OAuth persistent grants

PingFederate 7.3 added an index (`EXPIRESIDX`) for the `expires` column in the `pingfederate_access_grant` database table. For information on adding this index to your existing table, see Reviewing database changes.

A new database table for OAuth persistent grant extended attributes

Starting with PingFederate 7.2 R2, a new database table needs to be created to support OAuth’s persistent grant extended attributes. The database scripts to create this table can be found in `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts/access-grant-attribute-<databaseServer>.sql` . See Reviewing database changesfor more information.

LDAP filter syntax checking

Starting with PingFederate 7.2, LDAP filters only allow spaces in matched-against values.

Examples

`(\|(sAMAccountName=${username})(employeeID=ID for ${username}))` is allowed; spaces in the matched-against value of “ `ID for ${username}` ” are valid.

`( \| (sAMAccountName=${username}) (employeeID=ID for ${username}) )` is not allowed because this filter contain spaces outside of matched-against values.

Invalid filters cause SSO runtime failures. Error messages logged to server.log include:

```
Caused by: javax.naming.NamingException: [LDAP: error code 87 - Expected a closing parenthesis...
```

```
Caused by: javax.naming.NamingException: [LDAP: error code 87 - Unexpected closing parenthesis found...
```

We recommend reviewing LDAP filters and removing spaces outside of matched-against values after upgrade.

HTML Form Adapter enhancement

Starting with version 7.1 R3, PingFederate tracks login attempts in the HTML Form Adapter. When the number of login failures reaches the Challenge Retries threshold defined in the adapter, the user is locked out for one minute. See HTML Form Adapter for more information.

A new index (`CLIENTIDIDX`) in the database table for OAuth persistent grants

PingFederate 7.1 R3 added an index (`CLIENTIDIDX`) for the `client_id` column in the `pingfederate_access_grant` database table. For information on adding this index to your existing table, see Reviewing database changes.

Requested (formerly SAML) AuthN Context authentication selector process order changed

In releases prior to 7.1 R2, when the Requested AuthN Context Authentication Selector received a list of authentication contexts, it used the last context that it could match, rather than the first. However, both the SAML and OpenID Connect specifications treat an authentication context list as appearing in order of preference. To align the Requested AuthN Context Authentication Selector with these specifications, the selection order was changed in 7.1 R2. With this release, the selector will use the first authentication context it can match, rather than the last.

Multivalued LDAP attributes passed to outbound provisioning OGNL expressions

In releases before version 7.1, if an OGNL expression was used to populate a SaaS-partner field in outbound provisioning, only the first value of a selected multivalued LDAP attribute was used in the OGNL expression. As of PingFederate 7.1, this behavior was changed to use all values in the expression.

> **Note**
>
> If this new behavior conflicts with existing deployments, it can be reverted via the `supportMultiValuesFromDirectory` property located in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.provisioner.mapping.OgnlFieldMapper.xml` file.

OAuth clients reconfiguration

Neither the Upgrade Utility nor the platform-specific installers migrates OAuth clients that are created from PingFederate 6.5 through 7.0. Use any of the following interfaces to reconfigure your OAuth clients:

- The **Clients** window (**Applications > OAuth > Clients**) in the PingFederate administrative console.
- The `/oauth/clients` administrative API endpoint.

- The REST-based web service for OAuth client management at the `/pf-ws/rest/oauth/clients` and `/pf-ws/rest/oauth/clients/id` endpoints. This web service requires the client records to be stored in a database.

Note that PingFederate has been storing OAuth clients in XML files since version 7.1; these clients are migrated to the new installation. In addition, if you have configured PingFederate 6.8 (or a more recent version) to store OAuth clients in an external database, the new installation retains that configuration as well.

Upgrade considerations introduced in PingFederate 6.x

Cluster bind address required

Starting with PingFederate 6.11, the `pf.cluster.bind.address` property located in `<pf_install>/pingfederate/bin/run.properties` is required when running PingFederate in a cluster. The default value is `NON_LOOPBACK`.

Decryption and digital signing policy changes

Potential security vulnerabilities have resulted in the following changes to PingFederate as of version 6.11. In some cases, these might impact interoperability with partners:

- When acting as an SP and using the POST binding, PingFederate decrypts an assertion only when the SAML response has been signed. An unsigned SAML response that contains an encrypted assertion is rejected.

> **Note**
>
> Although strongly discouraged, this policy change can be reverted on a per-connection basis via the `EntityIdsToAllowAssertionDecryptionWithoutResponseSignature` list located in the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.profiles.sp.HandleAuthnResponse.xml` file.

- When acting as an IdP, PingFederate always signs a SAML response (even when the assertion is also signed) if it contains an encrypted assertion.

> **Note**
>
> Although strongly discouraged, this policy change can be reverted on a per-connection basis via the `EntityIdsToOmitResponseSignatureOnSignedEncryptedAssertion` list located in the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.profiles.idp.HandleAuthnRequest.xml` file.

- When acting as an IdP, PingFederate decrypts an encrypted NameID in an Attribute Query only when the request has been signed or the client has authenticated with basic or mutual TLS.

Key transport algorithm deprecated

Due to security risks associated with the RSA-v1.5 algorithm used for key transport, it is no longer available for new connections starting with PingFederate 6.11. Existing connections in which this algorithm is configured continue to support it. However, we recommend upgrading to use the newer RSA-OAEP algorithm. See Selecting an encryption certificate for SP connections and Choosing an encryption certificate (SAML 2.0) for IdP connections.

OAuth persistent grants expiration

When upgrading to PingFederate 6.8 later, all persistent grants for any existing OAuth deployments using an Oracle MySQL database will expire. To address this issue, the expires column in the `pingfederate_access_grant` table should be set to null prior to the upgrade. If necessary, contact Ping Identity support for assistance.

OAuth clients reconfiguration

Neither the Upgrade Utility nor the platform-specific installers migrates OAuth clients that are created from PingFederate 6.5 through 7.0. Use any of the following interfaces to reconfigure your OAuth clients:

- The **Clients** window (**Applications > OAuth > Clients**) in the PingFederate administrative console.
- The `/oauth/clients` administrative API endpoint.
- The REST-based web service for OAuth client management at the `/pf-ws/rest/oauth/clients` and `/pf-ws/rest/oauth/clients/id` endpoints. This web service requires the client records to be stored in a database.

Note that PingFederate has been storing OAuth clients in XML files since version 7.1; these clients are migrated to the new installation. In addition, if you have configured PingFederate 6.8 (or a more recent version) to store OAuth clients in an external database, the new installation retains that configuration as well.

OGNL library upgraded

The OGNL library has been upgraded in version 6.4. If you use OGNL expressions in versions prior to 6.4, we recommend retesting the expressions using the PingFederate administrative console or runtime tests.

Upgrading PingFederate installations

When upgrading your PingFederate installation on Linux, you must use the Upgrade Utility. On Windows, you can use either the Upgrade Utility or the installer.

Before you begin

- Read Upgrading PingFederate for an overview of the upgrade process.
- Ensure that you are signed on to your server with appropriate privileges to install and run an application.

About this task

Choose the relevant procedure for your system to upgrade PingFederate.

> **Tip**
>
> If you are upgrading a clustered PingFederate environment, start with the console node, and then upgrade the engine nodes.

> **Important**
>
> All nodes must use the same version of PingFederate.

Upgrading on Linux

Upgrading PingFederate on Linux systems

You can use the Upgrade Utility on Linux servers to upgrade to the current version of PingFederate.

About this task

The Upgrade Utility copies all relevant sources of your current installation into the new target directory of your choice. It doesn't change the current installation.

The Upgrade Utility migrates the existing versions of all PingFederate plugins by default. You can use the `-c` command line parameter to override the default behavior and install the latest versions of each plugin.

Upgrade results are contained in the `<pf_install_target>/pingfederate/upgrade/log/upgrade.log` file.

To use the Upgrade Utility on Linux servers to upgrade to the current version of PingFederate:

Steps

1. Download the latest version of the PingFederate Server distribution `.zip` archive from the Ping Identity website.
2. Extract the `.zip` to the new target directory of your choice, `<pf_install_target>`.
3. Stop PingFederate.
4. On the command line, change directory to `<pf_install_target>/upgrade/bin` and execute the following command:

```
./upgrade.sh  <pf_install_source> [-l] [<newLicense>] [-c] [--release-notes-reviewed]
```

where: `<pf_install_source>` ::

The full or relative path of the base `pingfederate` directory where the existing PingFederate software is installed.

> **Note**
>
> The `pingfederate` directory must have the name "pingfederate" for the Upgrade Utility to function correctly.

`<newLicense>`

The optional path and file name of the license to use for the upgraded PingFederate version.

> **Note**
>
> If your current license is valid, the Upgrade Utility automatically copies it from the source installation to the target installation, and you don't need to specify the `<newLicense>` parameter.
> If your license isn't valid, obtain a valid license file and specify its path and file name for this parameter.

`-c`

The optional parameter to run the tool in custom mode, which allows you to override newer default security settings (if any) and to upgrade to the newest version of each installed plugin.

`--release-notes-reviewed`

An optional parameter that indicates that you've already reviewed the release notes. This parameter prevents prompts during the upgrade that ask if you've read the release notes and the upgrade considerations.

The command prompt displays messages indicating upgrade progress. The process is complete when the following message appears:

```
Upgrade completed with  [N]  errors and  [N]  warnings
```

If there are errors, scroll up the command window to see them and then correct the indicated problems. Errors during the upgrade should be rare but might include problems such as missing or malformed configuration files in the source installation. The messages are also logged to the `upgrade.log` file in the Upgrade Utility base directory.

1. If you're using an older version of the Amazon Web Services (AWS) CloudHSM client:

    1. Update the CloudHSM client and the CloudHSM Software Library for Java to a supported version and restart the client.

        For more information, see System requirements.

    2. Copy `<pf_install_target>/pingfederate/lib-ext/pf-aws-cloud-hsm-wrapper.jar` to the `JAVA_HOME/jre/lib/ext` directory.

    3. Copy all of the files in the `/opt/cloudhsm/java` and `/opt/cloudhsm/lib` directories to the `JAVA_HOME/jre/lib/ext` directory.

    If you're upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

Note

End users might experience disruptions while you upgrade your PingFederate environment.

1. If PingFederate is running as a service, reconfigure the PingFederate service:

    1. Edit the `<pf_install_target>/pingfederate/sbin/linux/pingfederate.service` systemd unit file (see step installing_and_uninstalling_pingfederate:pf_install_pf_service_on_linux_manually.adoc#pf_step_installPingfederateSys in Installing PingFederate > Installing on Linux manually).

    2. Edit the `<pf_install_target>/pingfederate/sbin/linux/pingfederate` SysV initialization script (see step installing_and_uninstalling_pingfederate:pf_install_pf_service_on_linux_manually.adoc#pf_step_installPingfederateSys in Installing PingFederate > Installing on Linux manually).

    Start the new PingFederate installation.

If you're upgrading a clustered PingFederate environment, start the new PingFederate instance on the console node.

If you've configured single sign-on using OpenID Connect (OIDC) as the console authentication scheme and set the endpoint settings back to your PingFederate environment, start the new PingFederate instance on the console node and one of the engine nodes.

1. Verify the new installation's version:
    1. Open the new installation's administrative console.
    2. On the toolbar, click the **Question Mark** icon. On the Help menu, click **About**.
    3. Verify that the pop-up shows the new version.
2. If you're upgrading a clustered PingFederate environment:
    1. Start the new installation on each engine node, and then ensure all nodes are shown on the **System > Server > Cluster Management** page.
    2. On the **Cluster Management** page, click **Replicate Configuration**.

1. Although the upgrade utility automatically merges, migrates, and copies the language packs' `.properties` files into the upgraded PingFederate installation, verify the language packs in the upgrade installation by looking at the `.properties` files located in the upgraded `<pf_install_target>/pingfederate/server/default/conf/language-packs` directory.
    - Standard `.properties` files include `pingfederate-email-messages.properties`, `pingfederate-messages.properties`, and `pingfederate-sms-messages.properties`. During the upgrade, these files are migrated and merged into the upgraded PingFederate installation.
    - Localized `.properties` files (for example, `pingfederate-messages_fr_CA.properties`), are also migrated and merged into the upgraded PingFederate installation.
    - If the PingOne MFA or PingOne Protect integration kit was installed on PingFederate, you must manually migrate its `.properties` file after the upgrade.
    - All other `.properties` files in `<pf_install_target>/pingfederate/server/default/conf/language-packs` that don't fit the previous criteria are copied (not merged) into the upgraded PingFederate installation.

Next steps

After upgrading PingFederate, perform the Post-upgrade tasks.

Upgrading on Windows with the Upgrade Utility

Upgrading PingFederate on Windows using the Upgrade Utility

You can use the Upgrade Utility on Windows servers to upgrade to the current version of PingFederate.

About this task

The Upgrade Utility copies all relevant sources of your current installation into the new target directory of your choice. It doesn't change the current installation.

Upgrade results are contained in the `<pf_install_target>\pingfederate\upgrade\log\upgrade.log` file.

> **Note**
>
> If you used the server distribution `.zip` archive to install the previous version of PingFederate on Windows, you must use the Upgrade Utility to upgrade to the current version of PingFederate.

Steps

1. Download the latest version of the PingFederate Server distribution `.zip` archive from the Ping Identity website.

   > **Note**
   >
   > The distribution `.zip` archive is identical for both Windows and Linux.

2. Extract the `.zip` to the new target directory of your choice, *<pf_install_target>*.

3. Stop PingFederate.

4. At the command prompt, change the directory to `<pf_install_target>\pingfederate\upgrade\bin` and enter the following command:

   ```
   upgrade  <pf_install_source>  <newLicense>] [-c] [--release-notes-reviewed]
   ```

   where: `<pf_install_source>` ::

   The full or relative path of the base `pingfederate` directory where the existing PingFederate software is installed.

> **Note**
>
> The `pingfederate` directory must have the name "pingfederate" for the Upgrade Utility to function correctly.

`<newLicense>`

The optional path and file name of the license to use for the upgraded PingFederate version.

> **Note**
>
> If your current license is valid, the Upgrade Utility automatically copies it from the source installation to the target installation, and you don't need to specify the `<newLicense>` parameter.
> If your license isn't valid, obtain a valid license file and specify its path and file name for this parameter.

`-c`

The optional parameter to run the tool in custom mode, which allows you to override newer default security settings (if any) and to upgrade to the newest version of each installed plugin.

`--release-notes-reviewed`

An optional parameter that indicates that you've already reviewed the release notes. This parameter prevents prompts during the upgrade that ask if you've read the release notes and the upgrade considerations.

The command prompt displays messages indicating upgrade progress. The process is complete when the following message appears:

```
Upgrade completed with  [N]  errors and  [N]  warnings
```

If there are errors, scroll up the command window to see them and then correct the indicated problems. Errors during the upgrade should be rare but might include problems such as missing or malformed configuration files in the source installation. The messages are also logged to the `upgrade.log` file in the Upgrade Utility base directory.

+ If you're upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

> **Note**
>
> End users might experience disruptions while you upgrade your PingFederate environment.

1. If PingFederate is running as a service, reinstall the service:

    1. Remove the existing PingFederate service.

       For more information, see Uninstalling PingFederate.

    2. Install the new PingFederate service.

       For more information, see Installing the PingFederate service on Windows manually.

    3. For a clustered PingFederate environment, reinstall the PingFederate service on all nodes.

   Start the new PingFederate installation.

If you're upgrading a clustered PingFederate environment, start the new PingFederate instance on the console node.

If you've configured single sign-on using OpenID Connect (OIDC) as the console authentication scheme and set the endpoint settings back to your PingFederate environment, start the new PingFederate instance on the console node and one of the engine nodes.

1. Verify the new installation's version:

    1. Open the new installation's administrative console.

    2. On the toolbar, click the **Question Mark** icon. On the Help menu, click **About**.

    3. Verify that the pop-up shows the new version.

2. If you’re upgrading a clustered PingFederate environment:

   1. Start the new installation on each engine node, and then go to **System > Server > Cluster Management** and ensure all nodes are shown.

   2. On the **Cluster Management** page, click **Replicate Configuration**.

1. Although the upgrade utility automatically merges, migrates, and copies the language packs' `.properties` files into the upgraded PingFederate installation, verify the language packs in the upgrade installation by looking at the `.properties` files located in the upgraded `<pf_install_target>\pingfederate\server\default\conf\language-packs` directory.

   - Standard `.properties` files include `pingfederate-email-messages.properties`, `pingfederate-messages.properties`, and `pingfederate-sms-messages.properties`. During upgrade, these files are migrated and merged into the upgraded PingFederate installation.

   - Localized `.properties` files (for example, `pingfederate-messages_fr_CA.properties`), are also migrated and merged into the upgraded PingFederate installation.

   - If the PingOne MFA or PingOne Protect integration kit was installed on PingFederate, you must manually migrate its `.properties` file after the upgrade.

   - All other `.properties` files in `<pf_install_target>\pingfederate\server\default\conf\language-packs` that don’t fit the previous criteria are copied (not merged) into the upgraded PingFederate installation.

Next steps

After upgrading PingFederate, perform the Post-upgrade tasks.

Upgrading on Windows with the installer

Upgrading PingFederate on Windows using the installer

If you used the PingFederate installer for Windows to install the previous version of PingFederate, you can use it to upgrade to the current version of PingFederate.

About this task

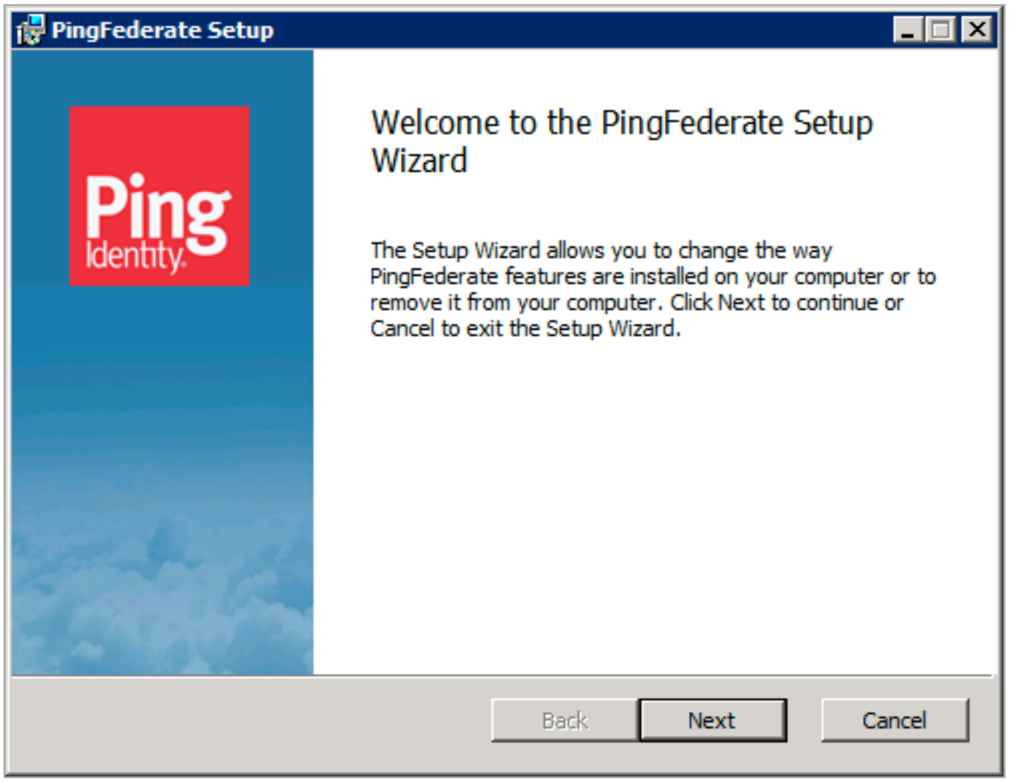
Upgrade results are contained in the `upgrade.log` file. If the upgrade succeeds, `upgrade.log` is located in `<pf_install_target>\pingfederate\upgrade\log`. If the upgrade fails, `upgrade.log` is located in `<pf_install_target>`.

The default location for `<pf_install_target>` is `C:\Program Files\Ping Identity`. You can change the location during the upgrade process.

> **Note**
>
> The PingFederate installer for Windows doesn't support custom mode. If you want to override the newer default security settings or upgrade the OpenToken Adapter, use the PingFederate Upgrade Utility.

Steps

1. Download the PingFederate installer for Windows from the Ping Identity website.
2. Double-click the `pingfederate-<version>.msi` file to begin the upgrade.
3. Follow the instructions in the PingFederate Setup Wizard to upgrade PingFederate.



Welcome to the PingFederate Setup Wizard

The Setup Wizard allows you to change the way PingFederate features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Errors are rare but might include problems, such as missing or malformed configuration files in the source installation. If the upgrade tool reports an error, review the error messages in the `upgrade.log` file.

Note

You can rerun the tool as many times as needed to correct any problems.

When the tool completes the upgrade, it automatically starts the new PingFederate installation.

+ If you're upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

Note

End users might experience disruptions while you upgrade your PingFederate environment.

1. Verify the new installation's version:
    1. Open the new installation's administrative console.
    2. On the toolbar, click the **Question Mark** icon. On the Help menu, click **About**.
    3. Verify that the pop-up shows the new version.
2. If you are upgrading a clustered PingFederate environment:
    1. Go to **System** > **Server** > **Cluster Management** and ensure all nodes are shown.
    2. On the **Cluster Management** page, click **Replicate Configuration**.

Next steps

After upgrading PingFederate, perform the Post-upgrade tasks.

Custom mode in the Upgrade Utility

The custom-mode feature in the Upgrade Utility (invoked with the `-c` option on the command line) allows you to override several default security settings. In addition, if the installed OpenToken Adapter is out of date, running the tool in custom mode allows you to replace the adapter with the latest version.

Security defaults

Using the security defaults should not cause significant issues for most PingFederate installations. The more recent default security settings include:

- Disabling weaker cipher suites for both the SUN and LUNA Java Cryptography Extension (JCE) in PingFederate version 6.2 and later. To see which cipher suites are commented out, choose yes (`y`) when prompted on whether to use the new defaults. After the upgrade is complete, refer to one of the following configuration files in the new installation's `<pf_install>/pingfederate/server/default/data/config-store` directory:
- `com.pingidentity.crypto.SunJCEManager.xml`
- `com.pingidentity.crypto.AWSCloudHSMJCEManager.xml`

- `com.pingidentity.crypto.LunaJCEManager.xml`
- `com.pingidentity.crypto.NcipherJCEManager.xml`
- `com.pingidentity.crypto.BCFIPSJCEManager.xml`

Adapter upgrade

Upgrading the OpenToken Adapter from an earlier version doesn't normally require any follow-on configuration changes.

- If your existing installation uses a version of the OpenToken Adapter earlier than 2.3, upgrading requires minor configuration modifications in the PingFederate console and redeployment of the agent configuration file.
- If you are upgrading from an OpenToken version earlier than 2.5.1, you should redeploy agent configuration files, if applicable, as well as any new agent libraries contained in recent versions of PingFederate integration kits and other plugins that use `OpenToken`.

Post-upgrade tasks

Confirm your new PingFederate installation configurations and settings.

After upgrading PingFederate, you might need to:

- Review administrative users
- Copy customized files or settings
- Review database changes
- Review the log configuration
- Migrate other components
- Reset files and variables for HSM
- Verify the new installation

You should also perform runtime tests to ensure the new PingFederate installation fulfills your existing use cases.

Reviewing administrative users

As of PingFederate 10.1, the use of expressions is enabled by default. Additionally, a new administrative role, Expression Admin, has been added.

When upgrading to the PingFederate 10.1 or later from a previous version, administrative users who were granted the Admin role in the earlier installation are granted the Expression Admin role automatically. You can achieve the same result by using the `/bulk/import` administrative API endpoint to bulk-import a configuration that was bulk-exported from PingFederate 10.0.

If preferred, administrators can disable the use of expressions by setting `evaluateExpressions` to `false` as described in Enabling and disabling expressions.

You can also go to **System** > **Server** > **Administrative Accounts** and remove the Expression Admin role from all Admin users. Doing this prevents Admin users from entering expressions into PingFederate if the `evaluateExpressions` element is set to `true` at a later time. For more information, see Administrative accounts.

Copying customized files or settings

After you upgrade PingFederate, you must copy files that were customized in the previous release to the current installation.

User-facing windows

If you modified any Velocity templates for user-facing windows, to preserve the customized user experience, some of your custom changes must be migrated manually to the new installation manually for each server node.

The templates are located in the `<pf_install>/pingfederate/server/default/conf/template` directory.

As of PingFederate 11.0, the Upgrade Utility migrates your modified Velocity HTML templates to the new installation. The default templates in the new installation that correspond to the modified templates are renamed with the following format: `<template_name>-default-<PF-version>.<ext>`.

> **Caution**
>
> Supporting CSS and image file names changed as of PingFederate 7.0. For each modified HTML template copied, add `.1` to the base name for each CSS file referenced in the header, for example, `<link rel="…" href="assets/css/window.1.css"/>`.
> Add `.1` to any references in the copied templates to the installed image files contained in the `assets/images` directory, for example, `<img src="assets/images/green_check.1.png"/>`.

Email notifications

If you modified the email notification templates before PingFederate 9.2, manually migrate your custom changes to the new HTML-based templates for each server node.

The plain text templates (`message-template-*.txt`) are located in `<pf_install>/pingfederate/server/default/conf` in the source installation. The new HTML-based templates are located in `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` with the same file naming convention but an `.html` file extension.

As of PingFederate 11.0, the Upgrade Utility migrates your modified Velocity HTML templates to the new installation. The default templates in the new installation that correspond to the modified templates are renamed with the following format: `<template_name>-default-<PF-version>.<ext>`.

Jetty or JBoss configuration

If you have modified any Jetty or JBoss settings that need to be carried forward, you must make the corresponding changes manually in the new PingFederate deployment.

If you are upgrading from PingFederate 6.9 or later, merge any changes in `<pf_install>/etc/jetty-runtime.xml` and `<pf_install>/etc/jetty-admin.xml` files into the corresponding files of the new deployment.

> **Note**
>
> If only the values of the `minThreads`, `maxThreads`, and `acceptQueueSize` parameters were modified in those Jetty files, you should set those values in the new deployment's `<pf_install>/bin/run.properties` file instead of merging them into the Jetty files.
> The advantage of setting those values in `run.properties` is that PingFederate can automatically merge `run.properties` customizations into new deployments, saving you time and preventing errors in future upgrades. For more information, see Configuring PingFederate properties.

If you are upgrading from PingFederate 6.0 through 6.8, first identify any changes made to the JBoss configuration, then make corresponding changes for the newer Jetty configuration.

For example, if you modified the `<pf_install>/pingfederate/server/default/deploy/jetty.sar/META-INF/jboss-service.xml` file before 6.9, identify the changes and make the same modifications at corresponding points in either the `jetty-admin.xml` or `jetty-runtime.xml` files located in the new Jetty configuration directory, `<pf_install>/pingfederate/etc`.

The size-limits.conf file

Before PingFederate 8.4.2, the `InterReqStateMgmtMapImpl.expiry.mins` setting in the `<pf_install>/pingfederate/server/default/conf/size-limits.conf` file defines the lifetime of the Adapter session-state data and Inter-request state information data sets.

Adapter session-state data

The state information, along with the associated attributes and any of their values, maintained or used by the adapters.

Inter-request state information

The state information between the redirects to complete a request.

PingFederate 8.4.2 and later splits the `InterReqStateMgmtMapImpl.expiry.mins` settings into two settings, one setting for each data type.

New settings	Data type	Default value in minutes
`InterReqStateMgmtMapImpl.expiry.mins.state.map`	Inter-request state information	30
`InterReqStateMgmtMapImpl.expiry.mins.attr.map`	Adapter session-state data	1440 (24 hours)

The new settings reduce the memory footprint of PingFederate by purging the inter-request state information after 30 minutes and retaining adapter session-state data during the day.

If you previously modified the value of the `InterReqStateMgmtMapImpl.expiry.mins` setting, when migrating your change to the latest version, adjust the value of the new settings based on your requirements.

Cross-origin resource sharing (CORS) support for OAuth endpoints

If you previously edited the `<pf_install>/pingfederate/etc/webdefault.xml` file to enable CORS support for OAuth endpoints, instead of updating the `webdefault.xml` file, define the allowed origins manually using the PingFederate administrative console after the upgrade.

For more information, see Configuring authorization server settings.

Configuration files in the config-store directory

If you added or replaced setting values in configuration files stored in the `<pf_install>/pingfederate/server/default/data/config-store` directory, the PingFederate upgrade tools copy these setting values to the new installation.

> **Note**
>
> The upgrade tools do not copy comments from the existing installation to the new installation.

If you removed a setting or a block of settings from a configuration file in the `config-store` directory, the upgrade tool preserves your changes by removing the setting or block of settings from the new installation and records the removals in its log file.

To re-add a setting or block of settings to the new installation, compare the configuration file found in the new installation to the file found in the product distribution `.zip` archive and make your changes.

Other configuration files

As of PingFederate 10.0, the upgrade process copies many files automatically. However, there are still some files that you must copy manually, which you can find in `<pf_install>/pingfederate/server/default/conf`.

The following files are copied automatically:

- Properties files with the `.conf` extension
- The `log4j2.db.properties` file
- The `jmx-remote-config.xml` file
- The `csd_configuration.yaml` file
- Non-default files located in the `template` directory

If you modified the default templates located in `<pf_install>/pingfederate/server/default/conf/template`, you must customize these templates in the new PingFederate installation.

If you modified versions of `tcp.xml`, `udp.xml`, and `log4j2.xml`, they are copied over intact. The default files are saved in the target directory with a different extension. To take advantage of the improvements in the default versions of these files, merge your changes into the current default files and then rename them appropriately.

> **Note**
>
> If you are upgrading from 8.0 or earlier, PingFederate might not start until you have merged your changes into the current default files because of JGroups errors.

Other files, such as `jmx.remote.access`, are not copied to the new installation automatically. To preserve any custom settings, create a backup of the current configuration files and merge your changes to the current files.

> **Important**
>
> If you previously customized Java virtual machine (JVM) options in the `run.bat` or `run.sh` files, instead of updating these files, manually merge your JVM options to the `<pf_install>/pingfederate/bin/jvm-memory.options` file. For more information, see Fine-tuning JVM options and memoryoptions and upgrade.

Reviewing database changes

Occasionally, PingFederate introduces database-related changes, such as adding a new table, modifying an existing table, or updating the connection pool library, for the purpose of product improvement.

Neither the Upgrade Utility nor the PingFederate installer for Windows migrates data maintained in the internal HSQLDB database or any external database. For instance, if outbound provisioning is enabled in the new PingFederate instance using the internal database, it's re-initialized from the provisioning source.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

If your PingFederate environment connects to one or more database servers, review the following topics and make changes accordingly:

- Provisioning datastore reset
- Enabling security enhancement in JDBC datastore queries
- An improved index in the database table for OAuth clients

Provisioning datastore reset

If you are upgrading from PingFederate 9.0.0 or 9.0.1, reset the provisioning datastore on the upgraded installation.

About this task

Use the `provmgr` command-line tool to reset the provisioning datastore on the upgraded installation.

The `provmgr` command-line tool is located in the `<pf_install>/pingfederate/bin` directory:

- Windows: `provmgr.bat`
- Linux: `provmgr.sh`

For more information about the `provmgr` command-line tool, see Outbound provisioning CLI.

Steps

1. To obtain a list of provisioning channel IDs, run the following command:

```
provmgr --show-channels
```

2. Reset the provisioning datastore for a given channel by its ID:

```
provmgr -c  <channel_id>  --reset-all
```

> **Note**
>
> If you have multiple provisioning channels, run the command for each channel. The order of the parameters doesn't matter.

Enabling security enhancement in JDBC datastore queries

Edit the `org.sourceid.common.SqlFilterManager.xml` file for stronger security protection in a JDBC datastore.

About this task

> **Note**
>
> If you are upgrading from PingFederate 8.4.4 or earlier, modify the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.SqlFilterManager.xml` file to enable the safeguard for JDBC datastore queries against backend SQL injection attacks.

Steps

1. Edit the `org.sourceid.common.SqlFilterManager.xml` file.
2. Set the `<item name="enableSqlFilters"/>` element value to `true`.

   Example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <config xmlns="http://www.sourceid.org/2004/05/config">
       <item name="enableSqlFilters">true</item>
   </config>
   ```

3. Save the file.
4. Restart PingFederate.
5. If you have a clustered PingFederate environment, push this change to all engine nodes:
   1. On the administrative console, go to **System > Server > Cluster Management**.
   2. Click **Replicate**.
6. Verify your use cases to make sure your search filters return the expected results.

An improved index in the database table for OAuth clients

PingFederate 8.4 added the `value` column to an existing index (`IDX_FIELD_NAME`) in the `pingfederate_oauth_clients_ext` table as a general improvement.

This information is applicable only to customers who configured PingFederate to store OAuth clients on a database server.

You must modify the index in your existing `pingfederate_oauth_clients_ext` table.

Although there is no alter-table script provided, you can derive the setup from the new table-setup scripts in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/sql-scripts/oauth-client-management-<databaseServer>.sql` file.

> **Note**
>
> If your initial installation of PingFederate was version 8.4 or later, you don't need to run this script.

Related links

- Configuring external databases for client storage.

Logging configurations

PingFederate uses log4j2 to write log messages.

If the configuration file for Log4j 2 (or Log4j) has been modified in the source installation, manually merge the configuration changes into the upgraded environment.

Merging custom logging configurations

About this task

The upgrade tools do not support automatic merging of customizations made to the existing logging configuration. Instead, these upgrade tools copy the modified `log4j2.xml` file to the new installation intact and rename the configuration file from the product `.zip` archive using the new PingFederate version number. Both configuration files are located in the same `conf` directory.

> **Note**
>
> If the Upgrade Utility or the PingFederate installer for Windows determines that the log4j2 configuration file (`<pf_install>/pingfederate/server/default/conf/log4j2.xml`) has changed since it was originally installed, new features are not activated.

To activate new features:

Steps

1. Review the new features by comparing the renamed log4j2 configuration file against the `log4j2.xml` file.
2. Modify the `log4j2.xml` file to suit your needs.
3. If you have a clustered PingFederate environment, repeat step 2 for all applicable PingFederate nodes in the cluster.

Migrating other components

Some custom and integrated components might require additional steps after upgrading PingFederate.

Updating the custom authentication selector

Through the use of the PingFederate SDK, you can create a custom authentication selector by implementing the `AuthenticationSelector` interface.

Most implementations return `AuthenticationSelectorContext.ResultType.CONTEXT` as the result type, which requires no further action after an upgrade.

If your implementation returns either:

- `AuthenticationSelectorContext.ResultType.ADAPTER_ID`, an Identity Provider (IdP) adapter instance ID
- `AuthenticationSelectorContext.ResultType.IDP_CONN_ID`, the connection ID of an IdP connection

You must update the descriptor instance of your custom authentication selector to call the `setSelectAuthnSourceResultType` method with an input of `true`. For each authentication policy path that ends with an instance of such custom authentication selector, you must ensure that its action is set to **Done**.

For more information, see the Javadoc for the `AuthenticationSelector` interface and the `AuthenticationSelectorDescriptor` class.

> **Tip**
>
> The Javadoc for PingFederate is located in the `<pf_install>/pingfederate/sdk/doc` directory.

Migrating to the integrated LDAP Username PCV

About this task

As of PingFederate 7.3, the integrated LDAP Username Password Credential Validator (PCV) can return additional attribute values upon successful validation.

If you have previously deployed the `LDAPExtendedAttributesPCV-<version>.jar` file from the PingID integration kit and created an instance of the LDAP PCV with Extended Attributes, migrate to the integrated LDAP Username PCV.

Steps

1. Create an instance of the integrated LDAP Username PCV:
   1. Go to **System > Data & Credential Stores > Password Credential Validators** and click **Create New Instance**.
   2. On the **Type** tab, enter the required information and select **LDAP Username Password Credential Validator** from the list.
   3. On the **Instance Configuration** tab, select an LDAP datastore from the list, enter a search base and a search filter, and select the scope of the search.

      > **Tip**
      >
      > You can reuse the information from the existing LDAP PCV with Extended Attributes instance.

    4. On the **Extended Contract** tab, enter `memberOf` in the **Extend the Contract** section, and click **Add.**
    5. On the **Summary** tab, review the setup and click **Done**.
    6. On the **Manage Credential Validator Instances** page, click **Save**.

2. In the configuration where the LDAP PCV with Extended Attributes instance is used, replace it with the newly created LDAP Username Password Credential Validator instance.

   For example, if you have created an instance of the PingID PCV (with integrated RADIUS server) instance and have selected an instance of the LDAP PCV with Extended Attributes as one of the delegate PCVs, remove the selection and add the newly created LDAP Username Password Credential Validator instance to the list.

3. After replacing the LDAP PCV with Extended Attributes instance, delete it from the **Password Credential Validators** page.
4. Remove the `<pf_install>/pingfederate/server/default/deploy/LDAPExtendedAttributesPCV-<version>.jar` file on all PingFederate servers.
5. Restart PingFederate on all PingFederate servers.

Related links

- Configuring the LDAP Username Password Credential Validator

Migrating to the integrated Username Token Processor

About this task

As of PingFederate 7.2, the Username Token Translator has been deprecated and replaced with an integrated Username Token Processor. Although the integrated Username Token Processor and the deprecated Username Token Translator can be simultaneously deployed, you should migrate it to the new token processor.

Steps

1. Go to **Identity Provider > Token Processors**.
2. To create an instance of the integrated Username Token Processor, click **Create New Instance**.
    1. On the **Type** page, select **Username Token Processor** from the list.

> **Tip**
>
> If you have multiple WS-Trust STS SP connections, you can reuse the same Username Token Processor instance or create additional instances of the token processors as needed.

3. Map the new token processor instance to the applicable WS-Trust STS SP connection on the **IdP Token Processor Mapping** page.

   Repeat this step if you have multiple WS-Trust STS SP connections.

4. Test your WS-Trust STS SP connections using the instance of the integrated Username Token Processor.
5. Remove the token processor instance of the deprecated Username Token Translator from all WS-Trust STS SP connections on the **IdP Token Processor Mapping** page.

6. If you have set up token translator mappings, create new entries to replace those using instances of the deprecated Username Token Translator, test the new mapping entries, and delete the entries that use instances of the deprecated Username Token Translator.
7. Delete all token processor instances of the deprecated Username Token Translator on the **Identity Provider > Token Processors** page.
8. Remove the `pf-username-token-translator-<version>.jar` file from the `<pf_install>/pingfederate/server/default/deploy` directory on all PingFederate servers.
9. Restart PingFederate on all PingFederate servers.

Related links

- Managing token processors
- Managing IdP token processor mappings
- Token translator mappings
- IdP protocol endpoints

Resetting files and variable for HSM

Update your environmental variable in the `<pf_install>server/default/data/ncipher-kmdata-local` directory to point to the new location.

If your PingFederate installation is configured in a clustered environment with Entrust nShield Connect, you must copy the `<pf_install>server/default/data/ncipher-kmdata-local` directory to the new installation manually and update the environmental variable `NFAST_KMLOCAL` to point to the new location.

Related links

- Integrating with Entrust nShield Connect HSM

Verifying the new installation

Verify the functionality of your new PingFederate installation.

PingFederate integration kits and custom solutions might introduce third-party dependencies that could trigger runtime errors. You should perform runtime tests, verifying that the previously deployed use cases, including any OGNL expressions, are functional in the new PingFederate installation.

Updating to the latest maintenance release

For PingFederate maintenance releases, you have the option to update your installation using the incremental update package. This in-place update method lets you replace and merge only the files that have changed.

Before you begin

- Ensure your installation is running PingFederate 11.3.x, where "x" represents an older maintenance release of 11.3.

- Make a backup copy of the PingFederate home directory.

About this task

You can incrementally update your PingFederate 11.3 installation to the latest 11.3 maintenance release. For example, if you are on PingFederate 11.3.1 and the latest maintenance release is version 11.3.2, you can use the in-place update method to update your installation to version 11.3.2.

The in-place update doesn't contain files from Ping Identity integration kits. You can upgrade an integration kit manually by downloading the latest kit from the PingFederate Downloads page's **Add-ons** tab and following the instructions provided by the kit's documentation.

> **Important**
>
> You cannot use the in-place update method to upgrade an older version of PingFederate, such as 10.3.2 or 11.1.4, to version 11.3.x. For those older versions, you must use the standard upgrade method for your platform described in Upgrading PingFederate.

> **Note**
>
> The in-place update method involves manual modification of PingFederate files. If you're not comfortable moving and editing these files, then instead use the standard upgrade method for your platform described in Upgrading PingFederate.

If your installation includes a cluster, perform the following procedure on each node starting with the administrative console node.

Steps

1. Go to the PingFederate Downloads page on the Ping Identity website.

2. In the **Maintenance Update** section, download the **In-place Update (ZIP)** file and extract its contents.

3. Stop PingFederate.

4. Copy the files in the in-place update's `pingfederate` directory and paste them into their corresponding locations in your current PingFederate installation, replacing the old files.

   For example, if the in-place update contains `pingfederate/server/default/lib/pf-protocolengine.jar`, copy the `pf-protocolengine.jar` file to the same location in your PingFederate installation: `<pf_install>/pingfederate/server/default/lib`.

5. Compare each file in the in-place update's `merge_required` directory with the version of the file in your current PingFederate installation and manually merge the changes into the file in your current installation.

   For example, if the in-place update contains `merge_required/pingfederate/server/default/conf/language-packs/pingfederate-messages.properties`, copy the changes in the new version into the `pingfederate-messages.properties` file in your PingFederate installation.

6. Start PingFederate.

Getting Started with PingFederate

This guide provides information about configuring PingFederate to deploy a secure Internet-identity platform, including single sign-on (SSO), based on the latest security and business standards.

Starting and stopping PingFederate

Depending on the application mode and the operating system, the steps to start, stop, or restart PingFederate vary.

If you install or upgrade PingFederate using its platform-specific installer, PingFederate configures to run as a service. You can stop and disable the service and run PingFederate as a console application.

If you install or upgrade PingFederate manually by using the PingFederate product distribution file or the Upgrade Utility in command line, you can run PingFederate as a console application or install the PingFederate service manually and run it as a service.

Starting and stopping on Windows

Starting and stopping PingFederate on Windows

Steps

- To start PingFederate on Windows, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Open a command prompt.
    2. Go to `<pf_install>/pingfederate/bin`.
    3. Run `run.bat`.
    4. Keep the command prompt open.

  - If PingFederate was installed as a Windows service:

    1. Go to **Control Panel** > **System and Security** > **Administrative Tools** > **Services**.
    2. Right-click on the PingFederate service and click **Start**.

- To stop PingFederate on Windows, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Locate the command prompt running PingFederate.
    2. Press CTRL+C to terminate PingFederate.

  - If PingFederate was installed as a Windows service:

    1. Go to **Control Panel** > **Administrative Tools** > **Services**.
    2. Right-click on the PingFederate service and click **Stop**.

- To restart PingFederate on Windows, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Locate the command prompt running PingFederate.
    2. Press CTRL+C to terminate PingFederate.
    3. When PingFederate stops, run `run.bat`.
    4. Keep the command prompt open.

  - If PingFederate was installed as a Windows service:

    1. Go to **Control Panel** > **Administrative Tools** > **Services**.
    2. Right-click on the PingFederate service and select **Restart**.

Starting and stopping on Linux

Starting and stopping PingFederate on Linux

Steps

- To start PingFederate on Linux, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Open a terminal window.
    2. Go to `<pf_install>/pingfederate/bin`.
    3. Run `run.sh`.
    4. Keep the terminal window open.
  - If PingFederate was installed as a service:

    1. Open a terminal window.
    2. Enter the system-dependent service command to start PingFederate.
- To stop PingFederate on Linux, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Locate the terminal window running PingFederate.
    2. Press CTRL+C to terminate PingFederate.
  - If PingFederate was installed as a service:

    1. Open a terminal window.
    2. Enter the system-dependent service command to stop PingFederate.
- To restart PingFederate on Linux, do one of the following:

  Choose from:

  - If PingFederate was installed as a console application:

    1. Locate the terminal window that is running PingFederate.
    2. Press CTRL+C to terminate PingFederate.
    3. When PingFederate stops, run `run.sh`.
    4. Keep the terminal window open.
  - If PingFederate was installed as a service:

    1. Open a terminal window.
    2. Enter the system-dependent service command to restart PingFederate.

Opening the PingFederate administrative console

The PingFederate administrative console provides a wizard-like interface in which you configure your federation use cases.

About this task

To open the administrative console:

Steps

1. Start PingFederate. See Starting and stopping PingFederate.

   In a clustered PingFederate environment, start PingFederate on the console node.

2. Start a web browser.

3. Go to https://<*pf_host*>:9999/.

> ⓘ **Note**
>
> For PingFederate 10.1 and earlier, the administrative console is accessed at https://<*pf_host*>:9999/pingfederate/app.
> <*pf_host*> is the network address of your PingFederate server. It can be an IP address, a host name, or a fully qualified domain name. It must be reachable from your computer.
> `9999` is the default value of the `pf.admin.https.port` property in the `run.properties` file.

Setting up PingFederate

After installing PingFederate, use the PingFederate setup wizard to configure the necessary initial settings.

About this task

The setup wizard guides you through the following steps:

1. Agree to the Terms and Conditions.

2. Enter the base URL used for communicating with your PingFederate environment.

3. Connect to PingOne. You can do this step later. For more information, see Creating connections to PingOne.

4. Upload your PingFederate license.

5. Create an administrator account.

6. Review and finish.

> ⓘ **Note**
>
> The wizard appears every time you open the PingFederate administrative console until you complete your initial setup.
> If you exit the wizard before completing the setup, none of your information is saved.

Steps

1. Start the PingFederate server. For more information, see Starting and stopping PingFederate

2. In your browser, go to https://<*pf_host*>:9999/pingfederate/app. The setup wizard opens.

> **Note**
>
> <*pf_host*> is the network address of your PingFederate server. It can be an IP address, a host name, or a fully qualified domain name. It must be reachable from your computer.
> `9999` is the default value of the `pf.admin.https.port` property in the `run.properties` file.

3. Follow the instructions in the wizard.

> **Note**
>
> Click **Back** to return to a previous step.
> Click **X** and close the browser tab to cancel your setup process.
> Click **Restart** to start again from the beginning.

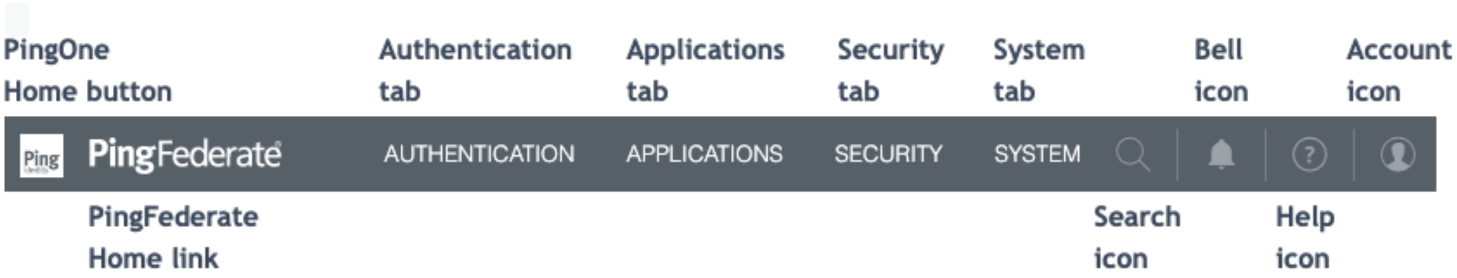
Result:

After you complete the setup wizard's steps, the PingFederate administrative console opens. You can continue configuring PingFederate.

PingFederate administrative console

The PingFederate administrative console provides pages and a wizard-like interface in which you configure your federation use cases.

The toolbar at the top of the administrative console provides controls mainly for navigating PingFederate. Learn more about navigating PingFederate in Navigation tabs and menus.



Toolbar control	Description
PingOne home button Ping	If PingFederate is integrated with PingOne, this button opens the PingOne unified admin console. ⓘ **Note** Configure the region and environment ID for your PingOne unified admin URL in the `run.properties` file. Learn more in Configuring PingFederate properties.
PingFederate home link	Opens the PingFederate administrative console's home page
Authentication tab	Opens the **Authentication** navigation page, which offers a menu pane and a shortcut pane
Applications tab	Opens the **Applications** navigation page, which offers a menu pane and a shortcut pane
Security tab	Opens the **Security** navigation page, which offers a menu pane and a shortcut pane
System tab	Opens the **System** navigation page, which offers a menu pane and a shortcut pane
Search icon 🔍	Opens a dialog box that lets you search for and open PingFederate pages
Bell icon 🔔	When a dot appears on the icon, clicking the icon opens a message about an important administrative issue. The messages also provide a link to the page where you can resolve the issue. ⓘ **Note** Some critical notifications appear on a banner above the toolbar.
Help icon ?	Opens the **Help** menu, which provides access to context-sensitive online help, a tour of the console, endpoint information, and information about your version and license
Account icon 👤	Lets you sign on to and off from your PingFederate account

Navigation tabs and menus

The PingFederate administrative console provides navigation tabs and menus. When you select a tab at the top of the console, the relevant menus appear on the left menu pane. When you select a menu, the menu items appear. When you select a menu item, the window with the same name appears. The menus and menu items depend on your permissions in PingFederate. Learn more about permissions in Administrative accounts.

Each navigation tab provides access to multiple menus on the left menu pane

Menus on the Authentication tab	Menus on the Applications tab	Menus on the Security tab	Menus on the System tab
• Integration • Policies • OAuth • Token Exchange	• Integration • OAuth • Token Exchange	• Certificate & Key Management • System Integration	• Data & Credential Stores • Server • OAuth Settings • External Systems • Monitoring & Notifications • Protocol Metadata

Navigation tabs, menus, and menu items in alphabetical order

Tabs	Menus	Menu items and window names
Applications	Integration	Adapter-to-Adapter Mappings
		Policy Contract Adapter Mappings
		SP Adapters
		SP Connections
		SP Default URLs
		Target URL Mapping
	OAuth	Access Token Management
		Access Token Mappings
		CIBA Request Policies
		Clients
		OpenID Connect Policy Management
	Token Exchange	Generator Groups

Tabs	Menus	Menu items and window names
		Processor Policies
		Token Generator Mappings
		Token Generators
		Token Translator Mappings
Authentication	Integration	Authentication API Applications
		IdP Adapters
		IdP Connections
		IdP Default URL
	OAuth	CIBA Authenticators
		IdP Adapter Grant Mapping
		Policy Contract Grant Mapping
		Resource Owner Credentials Grant Mapping
	Policies	Fragments
		Local Identity Profiles
		Policies
		Policy Contracts
		Selectors
		Sessions
	Token Exchange	STS Request Parameters
		Token Processors
Security	Certificate & Key Management	Certificate Revocation Checking
		OAuth & OpenID Connect Keys
		Partner Metadata URLs
		Signing & Decryption Keys & Certificates

Tabs	Menus	Menu items and window names
		SSL Client Keys & Certificates
		SSL Server Certificates
		System Keys
		Trusted CAs
	System Integration	Incoming Proxy Settings
		Redirect Validation
		Service Authentication
System	Data & Credential Stores	Active Directory Domains/Kerberos Realms
		Data Stores
		Identity Store Provisioners
		Password Credential Validators
	External Systems	CAPTCHA and Risk Providers
		Connect to PingOne for Enterprise
		Notification Publishers
		PingOne Connections
		SMS Provider Settings
	Monitoring & Notifications	Runtime Notifications
		Runtime Reporting
	OAuth Settings	Authorization Server Settings
		Client Registration Policies
		Client Settings
		Scope Management
	Protocol Metadata	Attribute Requester Mapping
		File Signing

Tabs	Menus	Menu items and window names
		Metadata Export
		Metadata Settings
		SP Affiliations
	Server	Administrative Accounts
		Cluster Management
		Configuration Archive
		Extended Properties
		General Settings
		License
		Virtual Host Names

Customizing shortcuts

Shortcuts allow you to access specific locations within the PingFederate administrative console immediately. You can customize which shortcuts you want to use by opening the **Shortcuts** customization menu available on the PingFederate main window.

About this task

Shortcuts are available for all PingFederate roles. You can customize up to ten shortcuts.

Steps

1. Click the **Pencil** icon beside **Shortcuts** to open the Shortcuts customization menu.

   The currently configured shortcuts are shown in the upper pane.

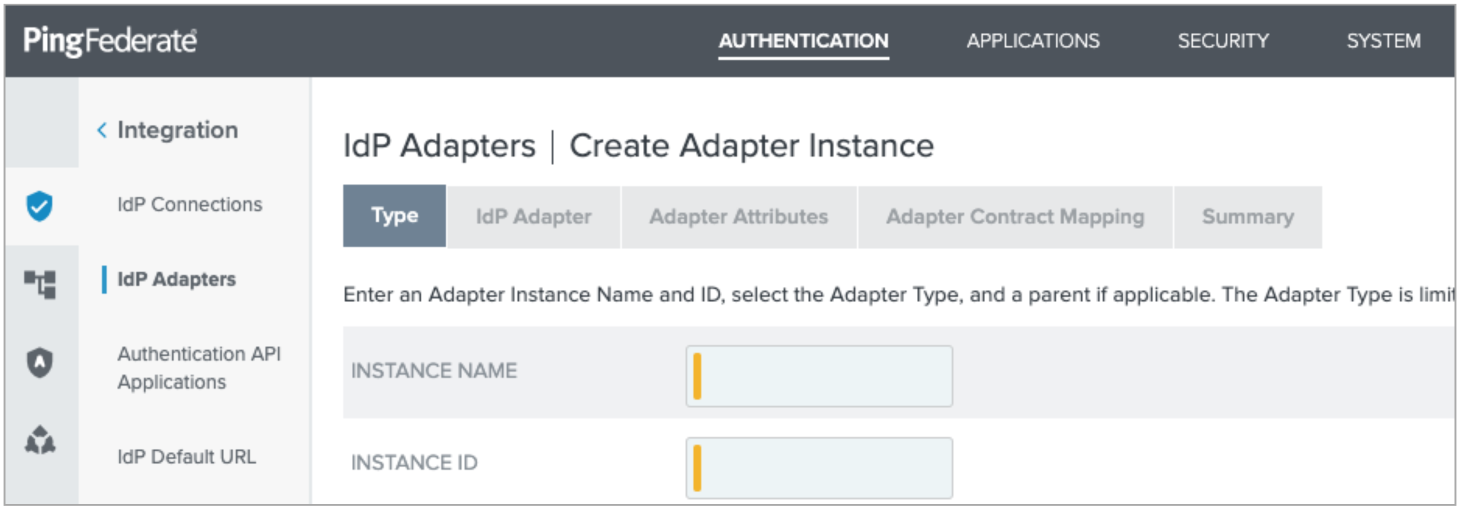
2. You can perform the following tasks.

   - To remove a current shortcut, click its icon in the upper pane. When you do this, the icon is displayed underneath **Recently Used**. Click the icon if you want to move it back to the list of current shortcuts.
   - To rearrange the current shortcuts, click and drag them to different positions in the upper pane.
   - To view the shortcuts that are available in every navigation section of the administrative console, click **Authentication**, **Applications**, **Security**, and **System**. Click a shortcut's icon or drag it into the upper pane to add it to the list of current shortcuts.
   - To restore the default shortcut settings, click **Restore Default**.

3. When you have finished customizing your shortcuts, click **Save**.

Tasks and steps

Each task consists of a series of tabs. Each tab consists of a sequence of steps. The tasks and tabs appear in the top portion of the page.



Enter an Adapter Instance Name and ID, select the Adapter Type, and a parent if applicable. The Adapter Type is limit

Sample tasks and steps

In this example, the primary task is managing one or more IdP adapter instances (**IdP Adapters**). The secondary task is creating an adapter instance (**Create Adapter Instance**). The current tab selects the type of adapter (**Type**). The subsequent tabs, which the administrator has not yet reached, are grayed out.

The administrator console displays a summary page at the end of every task, which offers the opportunity to review and make changes as needed.

Some steps provide buttons that branch to secondary tasks with multiple tabs. When the secondary tasks are complete, the administrative console returns to the primary task for the administrators to continue with the configuration.

> **Note**
>
> Clicking **Cancel** or **Done** discards all unsaved changes for the tabs shown in the current task and returns you to the page from which you accessed the task.

Example

When creating a connection to a partner, the administrator might need to create a new digital signing certificate. The administrative console provides a button to begin creating a new signing certificate. When the administrator completes the task, the administrative console returns to the primary task of creating a connection to a partner.

Console buttons

The buttons at the bottom of the administrative console change depending on where you are in the configuration process.

The following table describes these buttons.

Button	Description
Save	Saves changes for all tabs in the current task and returns to thewindow from which the task or tab was accessed. This button is available only when the **Save** operation is valid.
Done	Marks all steps as complete for a current task, but does not save the configuration because further tasks or steps are necessary. To save your changes, click **Save**, or continue the configuration until you see a **Save** button. When creating a new service provider (SP) or identity provider (IdP) connection, click **Save Draft**.
Save Draft	Saves a connection's draft configuration.
Cancel	Discards all changes and returns to the window from which the current task was accessed.
Previous	Returns to the previous tab.
Next	Proceeds to the next tab if all required steps are complete in the current tab.

> **Caution**
>
> Do not use the browser's **Back**, **Forward**, or**Refresh** buttons. Always use the navigation buttons in the PingFederate user interface, **Previous**, **Next**, or **Done**.

Admin console best practices

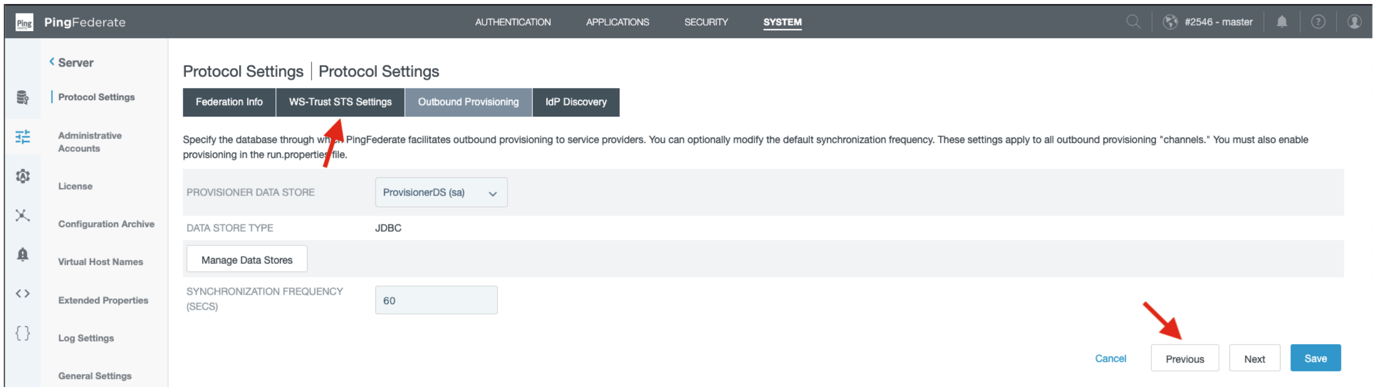
The admin console is a convenient way to configure and manage your PingFederate environment. However, you should keep the following best practices in mind while using the admin console so that you don't inadvertently create errors or corrupt your PingFederate configuration.

Don't use your browser's Back button

Using your browser's navigation buttons such as **Back** or **Reload**, can cause PingFederate to behave inconsistently.

If you need to go back to a previous menu, use the navigation buttons at the top or right side of the PingFederate menu.

To go back a step in a configuration workflow with multiple tabs, click the tab you want to visit, or click **Previous** at the bottom of the screen.



Protocol Settings | Protocol Settings

Specify the database through which PingFederate facilitates outbound provisioning to service providers. You can optionally modify the default synchronization frequency. These settings apply to all outbound provisioning "channels." You must also enable provisioning in the run.properties file.

PROVISIONER DATA STORE: ProvisionerDS (sa)

DATA STORE TYPE: JDBC

SYNCHRONIZATION FREQUENCY (SECS): 60

Don't use multiple browser tabs

Using PingFederate in multiple browser tabs could cause errors or inconsistencies.

Wait for a page to finish loading before going to another page

Going to another page before the current page finishes loading can cause inconsistent behavior. Even if you go to a page by mistake, allow it to load fully before going elsewhere.

One administrator at a time

You should allow only one administrative user to sign on to PingFederate at any one time. Because admins can create or change configurations, having multiple admins working in PingFederate at the same time risks causing conflicts.

> **Tip**
>
> To configure PingFederate to allow only one administrator at a time, set the `pf.console.login.mode` parameter in `run.properties` to `single` .

You can have one or more auditors in PingFederate at the same time as an admin. Auditors can see settings in the admin console, but they can't change them.

Don't replicate while an admin is working

If you're running PingFederate in a clustered environment, you should only replicate configurations to other nodes while no admins are working in the admin console. If an admin is creating or changing a configuration while replication is in progress, the replication might only carry over a portion of the configuration.

Clear your cache and cookies

If pages in PingFederate aren't loading at all, or it returns the **Something's not right** error page, trying clearing the cache and cookies in your browser. This can be particularly effective after an upgrade.

Third-party cryptographic solutions

PingFederate can use third-party hardware and software cryptographic solutions.

You can configure PingFederate to use a hardware security module (HSM) for cryptographic material storage and operations. When configured, private keys and their corresponding certificate are stored on the HSM. Related signing and decryption operations are processed there for enhanced security. By default, even in HSM mode, dynamic OAuth and OpenID Connect signing and decryption keys are generated and stored in the memory of PingFederate cluster nodes. To ensure continuity after a full cluster restart, the decryption keys are also persisted to disk, and encrypted there with PingFederate's active configuration encryption key. To ensure OAuth and OpenID Connect keys are instead stored on the HSM, you must enable static keys.

You can also integrate PingFederate with a third-party software cryptographic solution.

Hardware security modules

Typically, integrating with an HSM involves two steps:

1. Install and configure the HSM according to the manufacturer's documentation.

2. Follow the vendor-specific instructions to configure a new or existing PingFederate environment to use the HSM for key generation, storage, and operation.

> **Tip**
>
> Use HSM hybrid mode to store each relevant key and certificate on the HSM or the local trust store. This allows you to transition the storage of keys and certificates to an HSM without needing to deploy a new PingFederate environment to mirror the setup. For more information, see Transitioning to an HSM.

> **Note**
>
> Configuring PingFederate to use an HSM for cryptographic material storage and operations might impact performance. The level of impact depends on the performance of cryptographic functionality provided by the HSM and the network latency between PingFederate and the HSM. Consult with your HSM vendor for performance tuning if you plan to use an HSM in your PingFederate deployment.

Software cryptographic solution

PingFederate supports Bouncy Castle FIPS as the provider of its Java keystore and cryptographic operations.

Supported hardware security modules

PingFederate supports multiple configurations for secure material storage and processing.

PingFederate supports the following modules:

- AWS CloudHSM
- Thales Luna Network HSM
- Entrust nShield Connect HSM

Integrating with AWS CloudHSM

PingFederate supports multiple hardware security modules (HSMs), including Amazon Web Services (AWS) CloudHSM.

Before you begin

- Ensure that Java 8 or 11 is installed on the PingFederate server. For more information, see Installing Java.
- PingFederate must be deployed on one of the operating systems supported by both AWS CloudHSM and PingFederate. See System requirements and Supported platforms for the client SDKs in the AWS CloudHSM documentation for a list of mutually supported operating systems.

> **Note**
>
> As of version 11.2, PingFederate will no longer support the AWS CloudHSM Client SDK 3.

Steps

1. Request a crypto user (CU) account from your AWS CloudHSM administrator.

You need this account's username and password for your PingFederate installation.

2. Install and configure the AWS CloudHSM Java Cryptography Extension (JCE) provider for Client SDK 5. For more information, see Install and use the AWS CloudHSM JCE provider for Client SDK 5 in the AWS CloudHSM documentation.

> **Important**
>
> To ensure successful installation of the JCE provider, do not install the AWS CloudHSM client. If you are upgrading from PingFederate 11.1 or earlier, remove any existing CloudHSM client software.

3. Connect the Client SDK to the AWS CloudHSM cluster. For instructions, see .aws.amazon.com/cloudhsm/latest/userguide/cluster-connect.html//[Bootstrap the Client SDK^] in the AWS CloudHSM documentation.

4. Run the appropriate command for your operating system to ensure that keys are available to use even if a cluster not containing multiple HSMs is used:

   Choose from:

   - On Linux operating systems, run the `sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check` command.
   - On Windows operating systems, run the `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe --disable-key-availability-check` command.

5. If you plan to use elliptic curve (EC) keys for decryption, run the appropriate command for your operating system.

   Choose from:

   - On Linux operating systems, run the `sudo /opt/cloudhsm/bin/configure-jce --enable-ecdh-without-kdf` command.
   - On Windows operating systems, run the `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe --enable-ecdh-without-kdf` command.

6. Configure a new PingFederate installation on the network interconnected to the HSM.

> **Note**
>
> Go to the next step to integrate an existing PingFederate installation with your HSM.

7. To enable the Java interface and PingFederate integration, copy the `cloudhsm-jce-5.6.0.jar` file to the `pingfederate/startup` directory:

> **Note**
>
> - On Linux operating systems, the file location is `/opt/cloudhsm/java/cloudhsm-jce-5.6.0.jar`.
> - On Windows operating systems, the file location is `C:\Program Files\Amazon\CloudHSM\java\cloudhsm-jce-5.6.0.jar`.

8. If you are upgrading from PingFederate 11.1 or earlier, revert any previous changes to the `JAVA_HOME/jre/lib/security/java.security` file and remove the `pf-aws-cloud-hsm-wrapper.jar` and other files previously copied to `JAVA_HOME/jre/lib/ext`.

9. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

    1. Go to the `# Crypto provider services` section.

    2. Change the `jce.manager` and `certificate.service` service endpoints to the following:

    ```
    ...
    jce.manager=com.pingidentity.crypto.AWSCloudHSMJCEManager
    ...
    certificate.service=com.pingidentity.crypto.AWSCloudHSMCertificateServiceImpl
    ...
    ```

    > **Note**
    >
    > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

10. Update the `<pf_install>/pingfederate/bin/run.properties` file:

    1. Change the value of the `pf.hsm.mode` property from `OFF` to `AWSCLOUDHSM`.

    2. If you are setting up a new PingFederate installation, set the value of the `pf.hsm.hybrid` property to `false` to store newly-created or imported certificates on your HSM.

    3. If you are configuring an existing PingFederate installation, set the value to `true` for the flexibility to store each relevant key and certificate on the HSM or the local trust store.

    This allows you to transition the storage of keys and certificates to your HSM without deploying a new PingFederate environment. For more information, see Transitioning to an HSM.

11. Run:

    - The `<pf_install>/pingfederate/bin/hsmpass.sh` script on Linux operating systems.

    - The `<pf_install>/pingfederate/bin/hsmpass.bat` command on Windows operating systems.

12. Enter the password for the CU account that you requested in step 1 when prompted.

    This procedure securely stores the password for communication to the HSM from PingFederate.

13. If the username of the CU account is not `crypto_user`, update the `com.pingidentity.crypto.AWSCloudHSM.xml` file:

    1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.AWSCloudHSM.xml` file.

        The unmodified version of the `com.pingidentity.crypto.AWSCloudHSM.xml` file is shown in the following:

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <con:config xmlns:con="http://www.sourceid.org/2004/05/config">
        <con:item name="Partition">PARTITION_1</con:item>
        <con:item name="CryptoUser">crypto_user</con:item>
    </con:config>
    ```

    2. Replace `crypto_user` with the username of the CU account.

Example:

In the following example, the username of the CU account is `example_user`.

```
<?xml version="1.0" encoding="UTF-8"?>
<con:config xmlns:con="http://www.sourceid.org/2004/05/config">
    <con:item name="Partition">PARTITION_1</con:item>
    <con:item name="CryptoUser">example_user</con:item>
</con:config>
```

14. Repeat these steps on each node.

15. Start the new PingFederate server or restart the existing PingFederate server.

AWS CloudHSM operational notes

When using a hardware security module (HSM), some restrictions apply to PingFederate.

- PingFederate requires Java 8 or 11 for deployment.
- As an OpenID Provider, PingFederate can use static or dynamically-rotating keys to sign ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects. When using dynamically-rotating keys as part of the default configuration, the memory, not the HSM, stores short-term keys. The HSM can store static keys.
- Private keys are not exportable. When configured for use with the HSM, PingFederate disables administrative-console options for this feature. Only the public portion of generated keys is exportable.
- When using the Configuration Archive feature, any keys, certificates, or objects generated and stored on the HSM prior to saving a configuration archive must continue to exist unaltered when the archive is restored. In other words, the PingFederate user interface must execute any deletion or creation of objects on the HSM for proper operation.

  For example, you create and save objects A, B, and C to the HSM and create a data archive that contains references to those objects. If you delete object C and attempt to recover it through the data archive, PingFederate fails. Because the data archive contains a reference to the object and the object has been deleted from the HSM, you cannot use that data archive again.
- PingFederate limits cipher suites to those listed in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.AWSCloudHSMJCEManager.xml` file.

> **Note**
>
> If the CloudHSM client daemon disconnects during runtime, PingFederate automatically attempts to reconnect to the daemon.

Integrating with Thales Luna Network HSM

PingFederate supports multiple hardware security modules (HSMs), including Thales Luna Network HSMs.

Steps

1. Ensure that the PingFederate server has a supported Java virtual machine (JVM) installed.

   For more information, see Installing Java.

2. Install and configure your Thales Luna Network HSM, including the optional JSP package for Java, according to Thales's instructions.

   This includes creating a partition, creating a Network Trust Link (NTL), and assigning a client to a partition.

   1. Ensure the operation of the **`vtl verify`** command to indicate secure and proper communication with the HSM.

   2. Delete any unnecessary keys or objects created while testing communication to the HSM from the host running PingFederate.

   3. For your PingFederate installation, record the password used to open communication to the HSM through the NTL.

3. To enable the Java interface, copy the Luna library and program files to the Java installation as follows.

Operating system	Steps
Windows	Copy the `LunaAPI.dll` and `LunaProvider.jar` files from the `LUNA_HOME/jsp/lib` directory to the `<pf_install>/pingfederate/startup` directory.
Linux	Copy the `libLunaAPI.so` and `LunaProvider.jar` files from the `LUNA_HOME/jsp/lib` directory to the `<pf_install>/pingfederate/startup` directory.

   Prior to installing PingFederate, Thales provides sample Java applications to test that the Java HSM interface works. For more information, see the HSM documentation from Thales.

4. Update the `java.security` file in your Java environment by inserting `LunaProvider` after `SunJCE`, and then moving `SunRsaSign` and `SunEC` below `LunaProvider`. Ensure that the providers are numbered sequentially after your changes.

   - If the node uses Java 8, the `java.security` file is in the `JAVA_HOME/jre/lib/security` directory. Here's an example of an updated file for Java 8:

     ```
     # List of providers and their preference orders (see above):
     security.provider.1=sun.security.provider.Sun
     security.provider.2=com.sun.net.ssl.internal.ssl.Provider
     security.provider.3=com.sun.crypto.provider.SunJCE
     security.provider.4=com.safenetinc.luna.provider.LunaProvider
     security.provider.5=sun.security.rsa.SunRsaSign
     security.provider.6=sun.security.ec.SunEC
     security.provider.7=sun.security.jgss.SunProvider
     security.provider.8=com.sun.security.sasl.Provider
     security.provider.9=org.jcp.xml.dsig.internal.dom.XMLDSigRI
     security.provider.10=sun.security.smartcardio.SunPCSC
     ```

   - If the node uses Java 11, the `java.security` file is in the `JAVA_HOME/conf/security` directory. Here's an example of an updated file for Java 11:

```
# List of providers and their preference orders (see above):
security.provider.1=SUN
security.provider.2=SunRsaSign
security.provider.3=SunEC
security.provider.4=SunJSSE
security.provider.5=SunJCE
security.provider.6=com.safenetinc.luna.provider.LunaProvider
security.provider.7=SunJGSS
security.provider.8=SunSASL
security.provider.9=XMLDSig
security.provider.10=SunPCSC
security.provider.11=JdkLDAP
security.provider.12=JdkSASL
security.provider.13=SunPKCS11
```

5. On the network interconnected to the HSM, set up a new PingFederate installation.

> **Note**
>
> To integrate an existing PingFederate installation with your HSM, skip to the next step.

6. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

    1. Go to the `# Crypto provider services` section.

    2. Change the `jce.manager` and `certificate.service` service endpoints to the following:

    ```
    ...
    jce.manager=com.pingidentity.crypto.LunaJCEManager
    ...
    certificate.service=com.pingidentity.crypto.LunaCertificateServiceImpl
    ...
    ```

    > **Note**
    >
    > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

7. In `com.pingidentity.crypto.LunaPartitions.xml`, configure `DefaultPartitionSlotOrLabel` with the slot number or label associated with the HSM partition you created in [step1].

8. Update the `<pf_install>/pingfederate/bin/run.properties` file.

    1. Change the value of `pf.hsm.mode` from `OFF` to `LUNA`.

    2. To configure a new PingFederate installation, set the value of `pf.hsm.hybrid` to `false`. When set to `false`, the HSM stores newly created or imported certificates.

To configure an existing PingFederate installation, set the value to `true` for the flexibility to store each relevant key and certificate on the HSM or the local trust store. This allows you to transition the storage of keys and certificates to your HSM without deploying a new PingFederate environment. For more information, see Transitioning to an HSM.

9. From the `<pf_install>/pingfederate/bin` directory, run the `hsmpass.bat` batch file for Windows or the `hsmpass.sh` script for Linux.

    1. Enter the NTL password when prompted. For more information, see [step1].

       This procedure securely stores the password for NTL communication to the HSM from PingFederate.

> **Note**
>
> The Thales Luna Network HSM supports configuration in a high-availability group. For more information, see the Thales distributed-installation instructions. To properly synchronize data, ensure that the `HAOnly` property is enabled using the **`vtl haAdmin -HAOnly -enable`** command.

10. Repeat these steps on each node.

11. Start the new PingFederate server or restart the existing PingFederate server.

> **Important**
>
> Whenever you restart the Luna HSM, Thales recommends you also restart dependent processes such as PingFederate and all server nodes in a cluster.

SafeNet Luna Network HSM operational notes

When using a hardware security module (HSM), some restrictions apply to PingFederate.

- As an OpenID Provider, PingFederate can use static or dynamically-rotating keys to sign ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects. When using dynamically-rotating keys as part of the default configuration, the memory, not the HSM, stores short-term keys. The HSM can store static keys.

- Private keys are not exportable. When configured for use with the HSM, PingFederate disables administrative-console options for this feature. Only the public portion of generated keys is exportable.

- When running in FIPS 140-2 level 3 compliance, also called strict FIPS mode, private keys cannot be imported. In this mode, administrative-console options for this feature are disabled.

- When using the Configuration Archive feature, any keys, certificates, or objects generated and stored on the HSM prior to saving a configuration archive must continue to exist unaltered when the archive is restored. In other words, the PingFederate user interface must execute any deletion or creation of objects on the HSM for proper operation.

  For example, you create and save objects A, B, and C to the HSM and create a data archive that contains references to those objects. If you delete object C and attempt to recover it through the data archive, PingFederate fails. Because the data archive contains a reference to the object and the object has been deleted from the HSM, you cannot use that data archive again.

- PingFederate limits cipher suites to those listed in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.LunaJCEManager.xml` file.

Integrating with Entrust nShield Connect HSM

PingFederate supports multiple hardware security modules (HSMs), including Entrust nShield Connect HSM.

Steps

1. Ensure the PingFederate server has a supported Java virtual machine (JVM) installed.

   For more information, see Installing Java.

2. Install and configure your Entrust nShield Connect HSM client software.

   As part of the installation, install the optional Java Support (including KeySafe) and nCipherKM JCA/JCE provider classes components.

3. After installation, see the HSM documentation from Entrust to make your PingFederate server a client of an HSM server.

   > **Note**
   >
   > PingFederate supports both Operator Card Set (OCS) protected keys and module-protected keys.
   > For OCS, note the password. You need the password for your installation of PingFederate.
   > For module-protected keys, edit the `pingfederate/server/default/data/config-store/com.pingidentity.crypto.NCipherSettings.xml` file to add the following entries:
   >
   > ```
   > <con:item name="protect">module</con:item>
   > <con:item name="ignorePassphrase">true</con:item>
   > ```

4. To enable the Java interface, copy the `NFAST_HOME/java/classes/nCipherKM.jar` file to the `<pf_install>/pingfederate/startup` directory.

5. If you're upgrading from PingFederate 11.1 or earlier, revert any previous changes to the `JAVA_HOME/jre/lib/security/java.security` file and remove the `nCipherKM.jar` file previously copied to `JAVA_HOME/jre/lib/ext`.

6. Set up a new PingFederate installation on the network interconnected to the HSM.

   > **Important**
   >
   > Skip to the next step to integrate an existing PingFederate installation with your HSM.

7. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Crypto provider services` section.

   2. Change the `jce.manager` and `certificate.service` service endpoints to the following:

      ```
      ...
      jce.manager=com.pingidentity.crypto.NcipherJCEManager
      ...
      certificate.service=com.pingidentity.crypto.NcipherCertificateServiceImpl
      ...
      ```

      > **Note**
      >
      > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

8. Update the `<pf_install>/pingfederate/bin/run.properties` file.

    1. Change the value of `pf.hsm.mode` from `OFF` to `NCIPHER`.

    2. If you are configuring a new PingFederate installation, set the value of `pf.hsm.hybrid` to `false` to store newly created or imported certificates on your HSM.

    3. If you are configuring an existing PingFederate installation, set the value to `true`, which provides the flexibility to store each relevant key and certificate on the HSM or the local trust store. This capability allows you to transition the storage of keys and certificates to your HSM without the need to deploy a new PingFederate environment and to mirror the setup. For more information, see Transitioning to an HSM.

9. From the `<pf_install>/pingfederate/bin` directory, run the `hsmpass.bat` batch file for Windows or the `hsmpass.sh` script for Linux.

    Enter the Operator Card Set password when prompted. See [step2].

    This procedure securely stores the password for communication to the HSM from PingFederate.

10. If you're not using the default slot for OCS protection, specify the slot in `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.NCipherSettings.xml`.

11. If you are setting up a new or configuring an existing PingFederate cluster, repeat these steps on each node.

    When finished, use the following steps to replicate nShield data to the connected nodes in the cluster.

    1. On the console node, go to the `<pf_install>/pingfederate/server/default/data` directory and create a sub directory named `ncipher-kmdata-local`.

    2. Copy to the `ncipher-kmdata-local` directory all files from the `NFAST_KMDATA\local` directory, where `NFAST_KMDATA` is an environment variable created during the nShield Connect installation.

        For example, `NFAST_KMDATA` could be set to `C:\ProgramData\nCipher\Key Management Data`.

    3. Create a new environment variable named `NFAST_KMLOCAL` and set it to `<pf_install>/pingfederate/server/default/data/ncipher-kmdata-local`.

        > **Note**
        >
        > You must define this environment variable on all servers within the cluster.

    4. Restart the nShield Connect hardserver on all PingFederate servers in the cluster. For instructions on restarting the hardserver, see the HSM documentation from Entrust.

    5. Sign on to the PingFederate administrative console and go to **System > Server > Cluster Management**.

    6. To push the configuration changes, including the nShield data, to the engine nodes, click **Replicate Configuration**.

12. Start the new PingFederate server or restart the existing PingFederate server.

nShield Connect HSM operational notes

Some restrictions apply to PingFederate when using a hardware security module (HSM).

- PingFederate only supports Operator Card Set (OCS) protected keys. If you use a standard, non-persistent OCS, removing the card from the smart card reader causes the HSM to remove the protected keys from its memory. Requests will likely fail because almost all requests require cryptographic processing. To resume operations, insert the card into the smart card reader and then restart PingFederate.

  Alternatively, use a persistent OCS so that protected keys remain in memory even after the card is removed from the smart card reader. PingFederate will continue to process requests and to load keys and certificates from the HSM as needed. Until the card is inserted back into the HSM, the HSM will not support new key and certificate creation and storage. However, using a persistent OCS does not require a restart of PingFederate in this situation. For more information about persistent OCS, consult your HSM vendor.
- As an OpenID Provider, PingFederate can use static or dynamically-rotating keys to sign ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects. When using dynamically-rotating keys as part of the default configuration, the memory, not the HSM, stores short-term keys. The HSM can store static keys.
- Private keys are not exportable. When configured for use with the HSM, PingFederate disables administrative-console options for this feature. Only the public portion of generated keys is exportable.
- When running in FIPS 140-2 level 3 compliance, also called strict FIPS mode, private keys cannot be imported. In this mode, administrative-console options for this feature are disabled.
- When using the Configuration Archive feature, any keys, certificates, or objects generated and stored on the HSM prior to saving a configuration archive must continue to exist unaltered when the archive is restored. In other words, the PingFederate user interface must execute any deletion or creation of objects on the HSM for proper operation.

  For example, you create and save objects A, B, and C to the HSM and create a data archive that contains references to those objects. If you delete object C and attempt to recover it through the data archive, PingFederate fails. Because the data archive contains a reference to the object and the object has been deleted from the HSM, you cannot use that data archive again.
- PingFederate limits cipher suites to those listed in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.NcipherJCEManager.xml` file.

Supported software security package

PingFederate supports the Bouncy Castle FIPS provider software security package.

Bouncy Castle FIPS provider

In Bouncy Castle FIPS mode, all security-related cryptographic operations in PingFederate are handled by the Bouncy Castle FIPS security provider. Bouncy Castle FIPS is a FIPS 140-2 validated software cryptographic module. Operating in Bouncy Castle FIPS mode may be required if PingFederate is running as part of a FedRAMP-certified cloud service.

Third-party libraries deployed in PingFederate, such as JDBC drivers, are not guaranteed to operate in a FIPS-compliant fashion. When FIPS 140-2 compliance is a goal, you should confirm with the vendor before using any third-party libraries.

Plugins such as adapters and password credential validators need to be individually assessed for FIPS compliance. The FIPS status of a plugin is displayed in the Summary page inside its configuration. A warning is also logged on start-up for any configured plugins that are not FIPS-compliant or have not yet been assessed.

The integration of Bouncy Castle FIPS provider supports two phases:

- **Hybrid** to transition private keys from default keystore to the Bouncy Castle keystore.
- **Non-Hybrid** to start storing private keys only in the Bouncy Castle keystore.

Several properties in the `<pf_install>/pingfederate/bin/run.properties` file allow you to configure these phases as shown in the following table.

Phase	Properties

Hybrid	`pf.hsm.mode=BCFIPS`
	`pf.hsm.hybrid=true`

Non-Hybrid	`pf.hsm.mode=BCFIPS`
	`pf.hsm.hybrid=false`

You can run either Java 8 or 11 when integrating with the BCFIPS provider. The setup steps are the same for both environments.

> **Important**
>
> The only way to switch from BCFIPS mode back to non-BCFIPS mode is to roll back PingFederate with an archive.

Integrating Bouncy Castle FIPS providers

This procedure describes how to integrate PingFederate with Bouncy Castle FIPS provider when you are running either Java 8 or 11.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.
    1. Go to the `# Crypto provider services` section.
    2. Change the `jce.manager` and `certificate.service` service endpoints to the following:

        ```
        ...
        jce.manager=com.pingidentity.crypto.BCFIPSJCEManager
        ...
        certificate.service=com.pingidentity.crypto.BCFIPSCertificateServiceImpl
        ...
        ```

        > **Note**
        >
        > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

2. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
    1. Change the `pf.hsm.mode` property to `BCFIPS`.
    2. If you are setting up a new PingFederate installation, set the value of the `pf.hsm.hybrid` property to `false` to store newly created or imported certificates on your HSM.
    3. If you are configuring an existing PingFederate installation, set the `pf.hsm.hybrid` value to `true` for the flexibility to store each relevant key and certificate on the HSM or the local trust store.

    This allows you to transition the storage of keys and certificates to your HSM without deploying a new PingFederate environment. For more information, see Transitioning to an HSM.
3. On Linux systems, the Bouncy Castle FIPS-approved secure random number generator may drain a large amount of entropy during initial seeding. If available entropy becomes too low, the PingFederate server or bundled command-line tools may stall on startup for long periods of time. If this occurs, then you will likely need to integrate with a hardware random number generator or install an entropy-supplementing daemon like **`rngd`**.

Bouncy Castle operational notes

When using the Bouncy Castle FIPS provider, some restrictions apply to PingFederate.

- As an OpenID Provider, PingFederate can use static or dynamically rotating keys to sign ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects. When using dynamically rotating keys as part of the default configuration, the memory, not the BCFIPS key stores, stores short-term keys. The HSM can store static keys.
- PingFederate limits cipher suites to those listed in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.BCFIPSJCEManager.xml` file.

Server Clustering Guide

This section introduces the server clustering functionality of PingFederate.

Use this guide to learn the following concepts and tasks for deploying PingFederate server clusters:

- Overview of clustering
- Cluster protocol architecture
- Runtime state-management architectures
- Runtime state-management services
- Deploying cluster servers
- Deploying provisioning failover
- Configuration synchronization

Overview of clustering

Server clustering lets you define a single configuration for multiple PingFederate servers and address single sign-on (SSO) and single logout (SLO) requests as a single system.

PingFederate includes clustering features that allow a group of PingFederate servers to appear as a single system to browsers and partner federation servers. In this configuration, all client traffic normally goes through a load balancer, which routes requests to the PingFederate servers in the cluster.

Server clustering can facilitate high availability of critical services and increase both performance and overall system throughput. However, availability and performance are often at opposite ends of the deployment spectrum, requiring you to balance them according to your deployment goals

> ⓘ **Note**
>
> PingFederate provides separate failover capabilities specifically for outbound provisioning, which by itself does not require either load balancing or state management. For more information, see Deploying provisioning failover.

Running multiple maintenance versions

You can run multiple maintenance versions of PingFederate in a cluster. For example, a cluster can contain servers running 10.0.0, 10.0.1, and 10.0.2. However, a cluster cannot contain servers running multiple major or minor versions, such as 10.0.0 and 10.1.0.

When running multiple versions in a cluster, the following message appears at the top of the console: `The cluster is running more than one version of PingFederate. Visit the Cluster Management page to see the versions.` The Cluster management window displays the version of PingFederate running on each server node.

> ⓘ **Note**
>
> Running multiple versions of PingFederate in a cluster might cause inconsistencies in runtime behavior.

Clustering with multiple maintenance versions can reduce the upgrade burden by eliminating downtime. Running in mixed mode should be a temporary solution, letting you gradually update each node until all of them are running the same maintenance version.

> **Important**
>
> When upgrading the servers in the cluster, upgrade the administrative console first. The maintenance release might contain UI changes that need to be replicated throughout the cluster.

General cluster architecture

The cluster architecture has two layers:

Cluster-protocol layer

The cluster-protocol layer allows the PingFederate servers to discover a cluster, communicate with each other, detect and relay connectivity failures, and maintain the cluster as individual servers join and leave.

Runtime state-management services

The runtime state-management services communicate session-state information required to process SSO and SLO requests. PingFederate abstracts its runtime state-management services behind Java service interfaces, which enables PingFederate to use interface implementations without regard to underlying storage and sharing mechanisms. Abstracting also provides a well-defined point of extensibility in PingFederate. Depending on the chosen runtime state-management architecture, each service can share session-state information with a subset of nodes or all nodes.

Runtime state-management architectures

PingFederate supports both adaptive clustering and directed clustering. Adaptive clustering offers the benefits of scaling PingFederate horizontally with little or no configuration requirement. Directed clustering allows administrators to specify which runtime state-management service uses which architecture model.

Group-RPC oriented approach

The prepackaged state-management implementations use a remote-procedure-call (RPC) framework for reliable group communication in a variety of deployments, allowing PingFederate servers to share state information within a cluster.

Load balancing

Clustered deployments of PingFederate for SSO and SLO transactions typically require the use of at least one load balancer, fronting multiple PingFederate servers.

When a client accesses the load balancer’s virtual IP, the balancer distributes the request to one of the PingFederate servers in the cluster. Based on the configuration of the associated runtime-state management service, the processing server contacts other PingFederate servers through remote procedure calls as it processes SSO and SLO requests.

PingFederate does not automatically balance the traffic among the servers in the cluster. To avoid overloading individual servers in the cluster, you must manage SSO and SLO requests externally. Because each server can only handle a certain amount of traffic, see PingFederate Performance Tuning Guide to plan ahead.

How you configure the balancer to select the appropriate server can vary from simple to highly complex, depending on your deployment requirements. For specific balancing strategies, their strengths and weaknesses, as well as the impacts on PingFederate performance, see Runtime state-management architectures.

Load balancers can incorporate SSL/TLS accelerators or work closely with them. Because of the high computational overhead of the SSL handshake, you should terminate SSL/TLS on a dedicated server external to PingFederate for deployments in which performance is a concern. You can still use SSL between the proxy or balancer and PingFederate but as a separate connection.

Server modes

In a cluster, you can configure each PingFederate instance, or node, as either an administrative console or a runtime engine. Runtime engines, also known as engine nodes, service federated-identity protocol requests, while the console server, also known as the console node, administers policy and configuration for the entire cluster through the administrative console. A cluster can contain one or more engine nodes but only one console node.

> **Note**
>
> To successfully process SSO requests, you must set the administrative console node to run outside of the load-balanced group.

Cluster protocol architecture

PingFederate's cluster-protocol services manage discovery, cluster messaging, connectivity failure detection, membership, and merging of split clusters.

> **Important**
>
> Nodes in the cluster must be able to communicate with one another over both the cluster bind port and the cluster failure detection port. This communication requirement remains true regardless of the chosen cluster discovery method or runtime state-management architecture.

Cluster discovery

PingFederate supports two cluster discovery methods.

Static discovery

Static discovery is suitable for a small cluster with about five to six engine nodes. Configuration requires no external component. You must configure each node with at least one expected node in a cluster. In practice, the initial discovery list should contain all nodes known in advance in the cluster, including itself, to increase the likelihood of new members finding and joining the cluster.

Dynamic discovery

Dynamic discovery is well-suited for environments where traffic volume may spike and require additional resources during peak hours. Instead of configuring a static list of known nodes ahead of time, configure new nodes to pull cluster membership information from a centralized repository. Because safe storage and ready accessibility of the information by all nodes is crucial, PingFederate supports IAM roles for Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and OpenStack Swift. The dynamic discovery method requires only a one-time setup. Once configured, maintaining a static discovery list requires no coordination effort.

Regardless of the discovery method, as individual nodes join and leave the cluster, the cluster-protocol service synchronizes the new membership information across all nodes.

Failure detection

The failure detection mechanism detects network connectivity failures by establishing TCP connections with other nodes at their cluster failure detection ports and sending occasional network messages. When a node detects a failure, it propagates the condition to other nodes, sharing new membership information across the cluster.

> **Important**
>
> If you deploy any networking devices, such as a firewall, between nodes, you must configure them to allow inbound TCP connections to the cluster failure detection ports, and not to terminate these connections based on their potentially low volumes of network activities.

Runtime state-management architectures

This section provides an overview of runtime state-managment architecture options available for PingFederate.

Runtime state-management services distribute session-state information in the cluster, making it possible for multiple nodes to handle SSO and logout requests as a single system. In a cluster consisting of a large number of nodes, it might be desirable for performance reasons to share session-state information with only a subset of nodes. PingFederate supports two runtime state-management architectures: adaptive clustering and directed clustering.

Adaptive clustering

Adaptive clustering automatically distributes session-state information to multiple nodes. Administrators do not have to modify individual configuration files to specify which nodes should participate in tracking user sessions.

In essence, each session receives an address from within an internally-defined range. For redundancy, multiple nodes store each session. These nodes form a replica set. Any node that receives a request and must look up or store session-state information can do so by calculating the address of the session and reaching out to the corresponding replica set.

As individual nodes join and leave the cluster, adaptive clustering redistributes session-state information to maintain the replica set throughout the cluster.

The default size of a replica set is three, which provides redundancy in case two nodes fail and ensures that a single node's slow response time doesn't delay requests. The `replication.factor` setting is in the `<pf_install>/pingfederate/server/default/conf/cluster-adaptive.conf` file.

Enable adaptive clustering by setting the `pf.cluster.adaptive` property in the `run.properties` file to `true`. This is the default state in new installations. For upgrades, if such property is not found or is set to `false`, the system disables adaptive clustering and enables directed clustering instead. To enable or disable adaptive clustering, set the `pf.cluster.adaptive` property to `true` or `false` on each node and then restart PingFederate. The `run.properties` file is in the `<pf_install>/pingfederate/bin` directory.

Important

After making changes to the `cluster-adaptive.conf` and the `run.properties` files, you must manually repeat the changes to all nodes in the cluster. The configuration replication process does not push these files across the cluster. When you are finished, restart PingFederate to apply the changes.

Note

Adaptive clustering doesn't support the SAML 2.0 single logout (SLO) profile using the SOAP binding. If you've configured one or more SAML 2.0 connections to support SLO using SOAP, you must either share all nodes or designate state servers deployment strategies in directed clustering. Learn more in Directed clustering.

Other advanced settings

Fine-tune each runtime state-management service implementation separately by modifying a configuration file located in the `<pf_install>/pingfederate/server/default/conf` directory. After making changes in these files, you must apply the changes to all nodes in the cluster manually.

Note

The adaptive clustering concept isn't applicable to the Artifact-Message Persistence and Retrieval Service, which shares messages with any node that requests the message. As needed, you can modify other applicable properties, such as the `rpc.timeout` property. Learn more in Artifact-Message Persistence and Retrieval Service.

The following tables indicate the configuration file that applies to each implementation and the applicable properties. See the indicated sections for detailed information about each implementation.

Configuration file and service implementation

Configuration file	RPC-based service implementation
`cluster-account-locking.conf`	Account Locking Service
`cluster-artifact.conf`	Artifact-Message Persistence and Retrieval Service
`cluster-assertion-replay-prevention.conf`	Assertion Replay Prevention Service
`cluster-idp-session-registry.conf`	IdP Session Registry Service
`cluster-inter-request-state.conf`	Inter-Request State-Management (IRSM) Service

Configuration file	RPC-based service implementation
`cluster-session-revocation.conf`	Back-Channel Session Revocation Service
`cluster-sp-session-registry.conf`	SP Session Registry Service

Property description

Property	Description
`rpc.timeout`	How long, in milliseconds, this node waits before timing out unresponsive RPC invocations. The default value is `500`, or half a second.
`synchronous.retrieve.majority.only`	Indicates how many responses to wait for when making synchronous remote procedure calls. When set to `true`, this node waits for the majority of the local replica set to respond. When set to `false`, it waits for all recipients to respond. `true` is the default value. **Note** This property is not applicable to the Account Locking Service and not found in the `cluster-account-locking.conf` file.
`bulk.revoked.sris.timeout` (found only in the `cluster-session-revocation.conf` file)	A node downloads a full revocation list from another node during startup or when it rejoins a cluster after being disconnected from it, for example due to a temporary network issue. This setting determines the amount of time in milliseconds PingFederate waits before aborting the download and reporting a timeout error. The default value is `10000`, which is 10 seconds.
`read.local.only` (found only in the `cluster-session-revocation.conf` file)	Determines how PingFederate should process queries for revocation status. When set to `true`, PingFederate processes queries for revocation status locally. When set to `false`, the processing node pulls revocation status from other engine nodes in the cluster, subject to the **rpc.timeout** value. `true` is the default value. **Note** When adding a session to the revocation list, the processing node always propagates the information to all engine nodes in the cluster. Learn more in Back-Channel Session Revocation Service.

Note

When you have enabled adaptive clustering, PingFederate ignores other properties found in these configuration files—namely `preferred.node.indices` and `preferred.node.group.id`. The latter is only in the `cluster-idp-session-registry.conf` file.

Related links

- Deploying cluster servers

Multi-region support

PingFederate supports multi-region server clusters in adaptive clustering architecture.

When a cluster spans multiple regions, administrators can specify region identifiers for different groups of nodes. When regions are defined, any node that receives a request and must store session-state information can do so by sending the information to replica sets in both the local and remote regions. Requests that require read-only access to session-state information are answered locally for optimal performance.

As individual nodes in different regions join and leave the cluster, adaptive clustering redistributes session-state information within the region where changes in the cluster membership occur. This approach strikes a balance between minimizing the volume of session-state network traffic and improving the accuracy of session-state information across regions.

Cross-region support is enabled by default when you configure region identifiers in adaptive clustering environments. PingFederate provides cross-region support for the following functions:

- User session-state information maintained by the Inter-Request State-Management (IRSM) Service, the IdP Session Registry Service, and the SP Session Registry Service
- Assertion Replay Prevention Service
- Account Locking Service
- Replication, validation, and revocation of access tokens using the reference token data model

When cross-region support is disabled in individual areas, engine nodes only communicate session-state information to and from the local replica set. To improve the accuracy of session-state information, you can deploy a network traffic management solution to persist, or stick, user sessions so that each subsequent request from the same user is directed to the same set of nodes.

> **Note**
>
> To reduce cross-region network traffic, PingFederate does not normally replicate SSO transaction states to other regions. However, if DNS sends user requests to different regions during a single SSO transaction, the transaction will fail with the error `Unable to resume processing because saved state was not found for key` .
> To let PingFederate asynchronously replicate SSO transaction states to other regions, open the `cluster-adaptive.conf` file and change the value of `inter.group.replicate.transaction.state` to `true` .

OAuth access token management

PingFederate shares reference token information with a replica set when adaptive clustering is enabled. If region identifiers are defined, PingFederate shares reference token information among multiple replica sets across regions. Like other services, you can optionally override this default behavior by changing the `inter.group.replicate.reference.tokens` value in the `<pf_install>/pingfederate/server/default/conf/cluster-adaptive.conf` file .

When you disable cross-region support for access tokens using the reference token data model, PingFederate does not share reference token information across regions. As a result, PingFederate cannot de-reference, validate, or revoke a reference-style access tokens issued outside of its region. For this reason, we recommended switching to the self-contained token data model prior to disabling cross-region support for the reference token data model.

Related links

- Token models and management
- Configuring an access token management instance

Configuring multi-region support

Define region identfiers and configure cross-region settings for multi-region PingFederate server clusters.

Steps

1. To define a region identifier for a given node, update the `node.group.id` value in the `<pf_install>/pingfederate/server/default/conf/cluster-adaptive.conf` file, which is a per-server configuration.

   Example:

   For example, if you have five engine nodes in the West Coast and six engine nodes in the East Coast, you can update the `node.group.id` value to `W` for each of the West Coast nodes and `E` for each of the six nodes in the East Coast.

   1. Restart PingFederate after making changes to the `cluster-adaptive.conf` file.

      Once defined, the identifiers for all nodes are displayed on the **System > Server > Cluster Management** menu.

2. To configure cross-region support for individual areas, follow the inline instructions in the `cluster-adaptive.conf` file to update the relevant setting values.

Directed clustering

This topic provides an overview of manual configuration of PingFederate sever nodes through directed clustering.

Directed clustering allows administrators to manually specify which PingFederate nodes should participate in tracking user sessions. Most group Remote Procedure Call (RPC)-based service implementations make use of a preferred-nodes concept, which assigns each node a list of other nodes, identified by index, with which it shares session-state information.

Each service implementation is controlled separately by a configuration file located in the `<pf_install>/pingfederate/server/default/conf` directory. You must replicate any changes manually for each cluster node.

The following tables indicate the configuration file that applies to each implementation and the applicable properties.

> **Note**
>
> The Artifact-Message Persistence and Retrieval Service uses only the `rpc.timeout` setting.

Configuration file and service implementation

Configuration file	RPC-based service implementation
`cluster-account-locking.conf`	Account Locking Service
`cluster-artifact.conf`	Artifact-Message Persistence and Retrieval Service
`cluster-assertion-replay-prevention.conf`	Assertion Replay Prevention Service

Configuration file	RPC-based service implementation
`cluster-idp-session-registry.conf`	IdP Session Registry Service
`cluster-inter-request-state.conf`	Inter-Request State-Management (IRSM) Service
`cluster-session-revocation.conf`	Back-Channel Session Revocation Service
`cluster-sp-session-registry.conf`	SP Session Registry Service

Property description

Property	Description
`preferred.node.indices`	A comma-separated list of indices identifying the nodes with which this node shares session-state information for the associated service. If left blank, this node sends session-state information to all nodes in the cluster as it processes SSO and logout requests. The Artifact-Message Persistence and Retrieval Service and the Back-Channel Session Revocation Service do not support this parameter. Ignored when adaptive clustering is enabled. This property has no default value.
`preferred.node.group.id` (found only in the `cluster-idp-session-registry.conf` file)	An alphanumeric group ID for a subcluster. If specified, each subcluster must have a unique group ID. At startup, PingFederate validates that the group ID is not already registered in the cluster by another list of preferred nodes. If the validation fails, PingFederate aborts the startup process and exits. When the group ID is specified, the session identifier contains the information about the originating subcluster. This is helpful in deployments where PingFederate has been configured to manage authentication sessions on the **Authentication** > **Policies** > **Sessions** window. When an engine node receives a request to query and extend a session, it can route the request to the corresponding subcluster based on the session identifier value. If subclusters are configured without specifying group IDs, a request to query and extend a session is processed on the subcluster that received the revocation status request, which may be different from the subcluster where the session is being tracked. As a result, the session could reach the idle timeout sooner than expected. Ignored when adaptive clustering is enabled. This property has no default value.
`rpc.timeout`	How long, in milliseconds, this node waits before timing out unresponsive RPC invocations. The default value is `500`, or half a second.
`synchronous.retrieve.majority.only`	Indicates how many responses to wait for when making synchronous remote procedure calls. When set to `true`, this node waits for the majority of recipients to respond. When set to `false`, it waits for all recipients to respond. The default value is `true`.

Property	Description
`bulk.revoked.sris.timeout` (found only in the `cluster-session-revocation.conf` file)	Determines the amount of time (in milliseconds) PingFederate waits before aborting the download of revocation lists and reporting a timeout error. The default value is `10000`, or 10 seconds. **Note** A node downloads a full revocation list from another node during startup or when it rejoins a cluster after being disconnected from it.
`read.local.only` (found only in the `cluster-session-revocation.conf` file)	Determines how PingFederate processes queries for revocation status. When set to `true`, queries for revocation status are processed locally. When `false`, the processing node pulls revocation status from other engine nodes in the cluster (subject to the **rpc.timeout** value). **Note** When adding a session to the revocation list, the processing node always propagates the information to all engine nodes in the cluster. See Back-Channel Session Revocation Service for more information. The default value is `true`.

Preferred node indices

Configuring the `preferred.node.indices` property can reduce the network communications and memory footprint. However, transaction volume and distribution can affect the resulting configuration. For more information on performance tuning, see PingFederate Performance Tuning Guide.

Individual services within a single cluster deployment can use different preferred nodes, meaning you can set different values for the `preferred.node.indices` property for each service.

Using preferred nodes can translate into a variety of deployment configurations. The following sections discuss three primary strategies to consider for meeting your network requirements:

- Sharing all nodes
- Designating state servers
- Defining subclusters

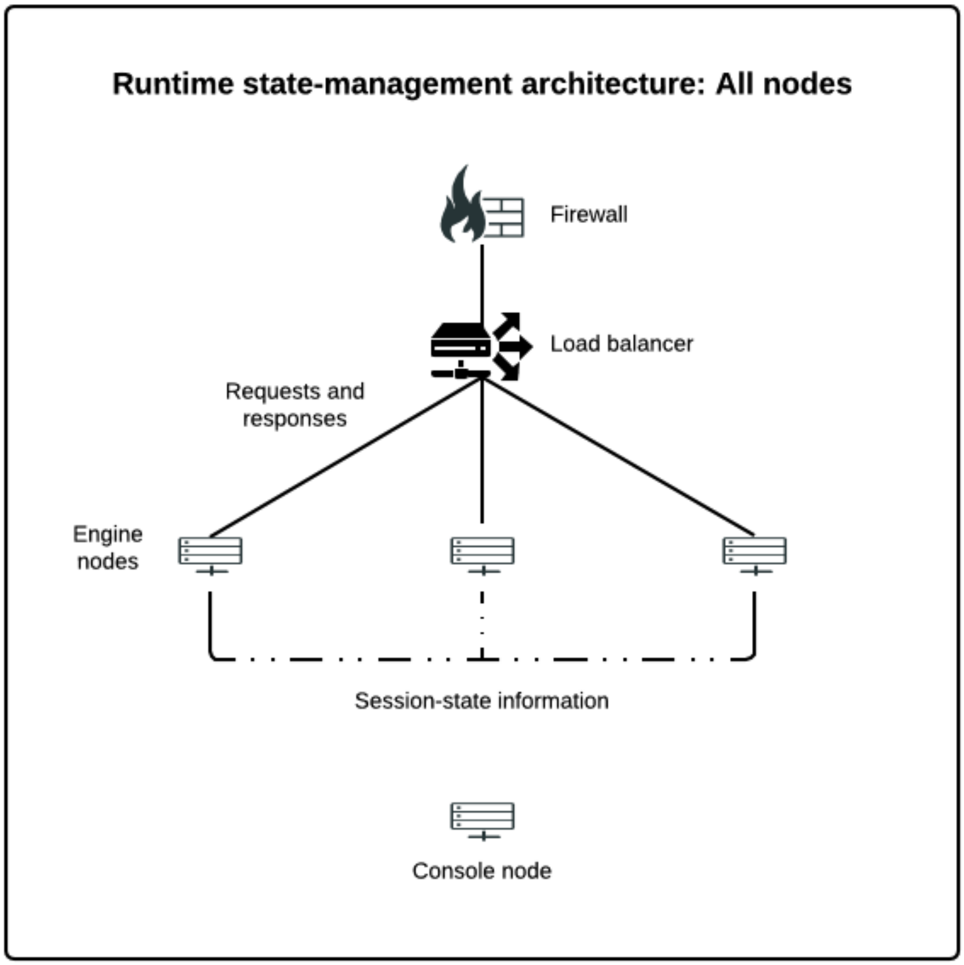
> **Note**
>
> It is possible to configure overrides for authentication-adapter processing based on the runtime node servicing a request. See Configuring the Cluster Node Authentication Selector for more information.

Sharing all nodes

The default directed clustering is the simplest deployment case, where all nodes are shared within a cluster.

Leaving the `preferred.node.indices` property blank in all cluster-configuration files results in a basic deployment case. An advantage of this approach is simplicity, including the option of using straightforward load-balancing strategies such as round robin. A disadvantage is that as additional nodes are added, the throughput improvement rate that clustering offers may decline as the state-replication overhead increases.

The following diagram illustrates this node-sharing approach. Requests in this deployment are directed to all nodes.


Runtime state-management architecture: All nodes
Firewall
Load balancer
Requests and responses
Engine nodes
Session-state information
Console node

Designating state servers

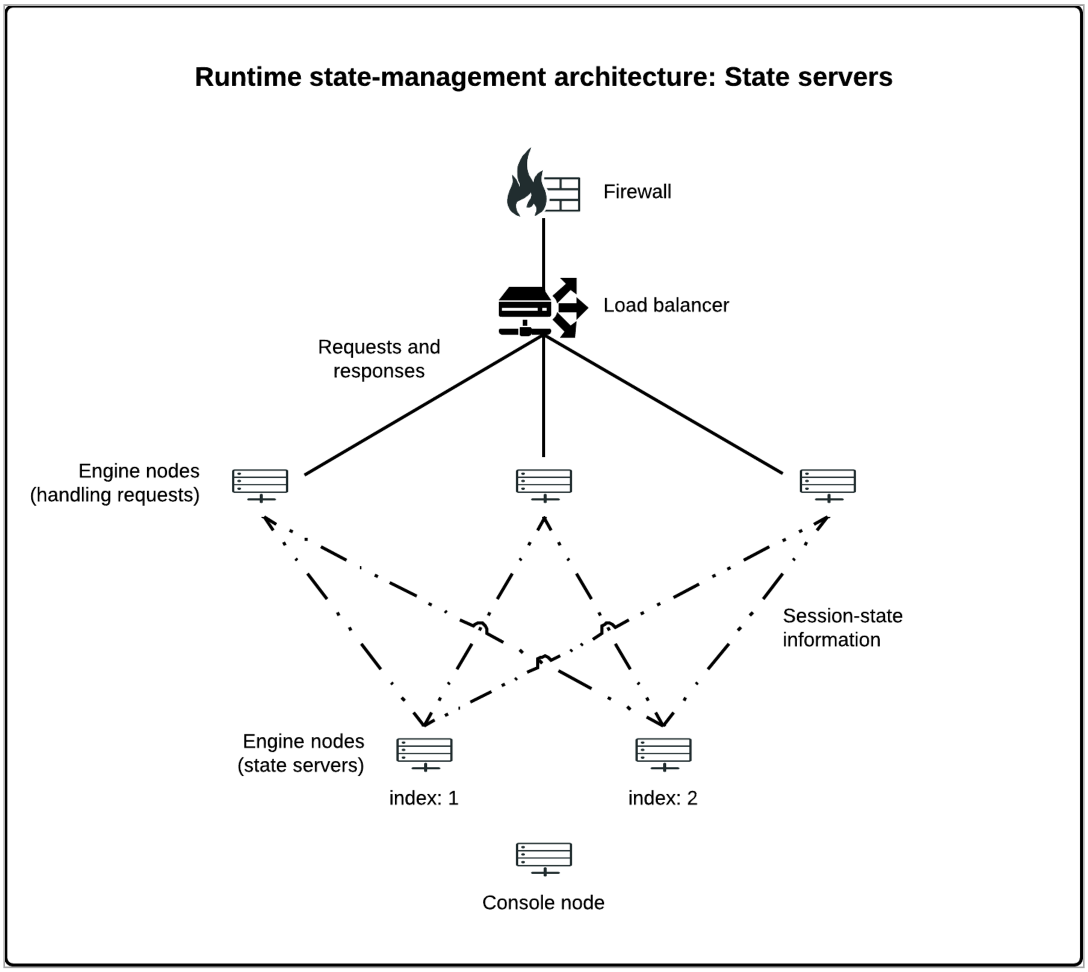
A state servers clustering deployment model improves scalabiltity by reducing communication between nodes.

You can select a few engine nodes as state servers. This deployment approach scales better than the all-nodes approach because additional nodes do not require connections to every existing node, they only require a connection between each server and each state server.

Configure this deployment by setting the `preferred.node.indices` of other servers in a group to those of the state servers. Configure the load balancer to isolate the state-server nodes from end-user traffic.

> **Note**
>
> The underlying cluster protocol still requires that all nodes are able to communicate with one another. The topology here is only an optimization for the runtime state-management services that support the concept of preferred nodes.

The following diagram illustrates the state-server approach.


Runtime state-management architecture: State servers
Firewall
Load balancer
Requests and responses
Engine nodes (handling requests)
Session-state information
Engine nodes (state servers)
index: 1
index: 2
Console node

In this example, the two state-server nodes have indices of 1 and 2. The `preferred.node.indices` property of the engine nodes handling requests would be `preferred.node.indices=1,2` .

Because the state servers are not processing transactions (based on the setup of the load balancer), the `preferred.node.indices` property for them is not used and can be left blank.

> **Note**
>
> When PingFederate acts as an OAuth authorization server (AS) and the access token management instance uses a reference token data model, the resource server (RS) must send a request to PingFederate to de-reference the access token for the corresponding identity and security information. Because the OAuth clients and the RS send their requests separately, PingFederate shares reference token information among all engine nodes despite any state server or subcluster setup.

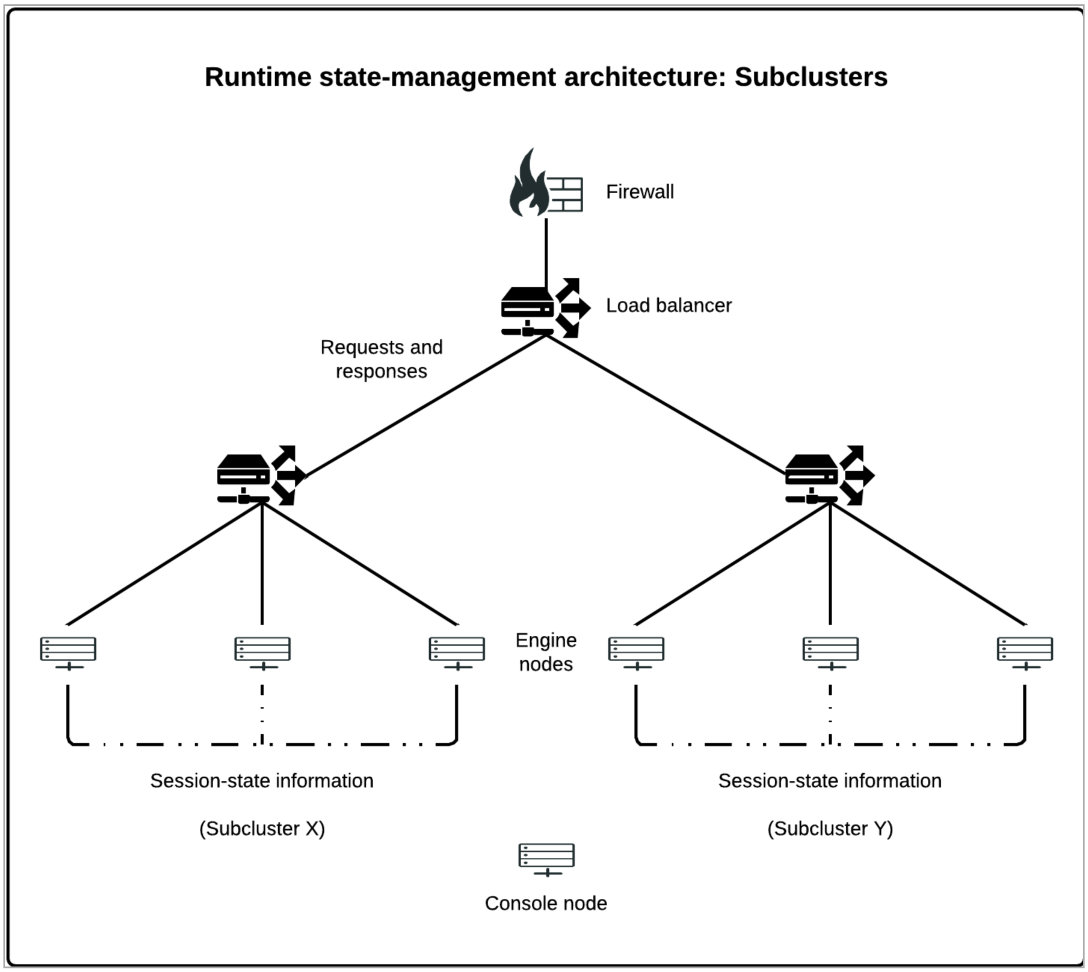
Defining subclusters

Subclustering improves efficient scaling by limiting session-state communication to other nodes within a subcluster.

Node indices can be configured to divide a cluster into subgroups, or subclusters, of a few nodes each. Using this configuration, each node in a subcluster shares session-state information only with other members of the subcluster. This approach requires a network traffic management solution to persist, or stick, user sessions so that each subsequent request from the same user is directed to the same set of nodes.

> **Note**
>
> The underlying cluster protocol still requires that all nodes are able to communicate with one another. The topology here is only an optimization for the runtime state-management services that support the concept of preferred nodes. Additionally, this architecture does not support OpenID Connect back-channel logout or the SAML 2.0 single logout (SLO) profile using the SOAP binding. If you need to use either of these capabilities, you must choose between sharing all nodes and designating state servers deployment strategies in directed clustering.
> This architecture also does not support the capability to revoke sessions after password change or reset. If you are using this capability, you are limited to the sharing all nodes and designating state servers deployment strategies.

The following diagram illustrates the subcluster approach.


Runtime state-management architecture: Subclusters
Firewall
Load balancer
Requests and responses
Engine nodes
Session-state information
(Subcluster X)
Session-state information
(Subcluster Y)
Console node

In this example, the `preferred.node.indices` property of each server in the cluster lists the indices of all nodes in its subgroup (including itself). Requests are directed to all nodes but the load balancer directs user sessions to the same subcluster.

ⓘ **Note**

When PingFederate acts as an OAuth authorization server (AS) and the access token management instance uses a reference token data model, the resource server (RS) must send a request to PingFederate to de-reference the access token for the corresponding identity and security information. Because the OAuth clients and the RS send their requests separately, PingFederate shares reference token information among all engine nodes despite any state server or subcluster setup.

Runtime state-management services

Runtime state-management services are a collection of interfaces defining the contract that PingFederate uses to manage session states for each service.

The `<pf_install>/pingfederate/server/default/conf/service-points.conf` file specifies the implementation of each service. You only need to modify this file if you want to customize the way services are handled in a cluster. You must manually replicate your changes in each cluster node.

By default, the proxies listed in `service-points.conf` select the best implementation for each service, based on the operational mode of the server. For example, an in-memory-only implementation applies if the server is in standalone mode, and a group RPC-based implementation applies to servers in clustered mode. These proxies are provided for convenience. You can specifically designate your desired implementation for each service, as described in the following sections.

- Inter-Request State-Management (IRSM) Service
- IdP Session Registry Service
- SP Session Registry Service
- LRU memory management schemes
- Assertion Replay Prevention Service
- Artifact-Message Persistence and Retrieval Service
- Back-Channel Session Revocation Service
- Account Locking Service
- Other services

Configuration files for the services are located in the `<pf_install>/pingfederate/server/default/conf` directory.

Inter-Request State-Management (IRSM) Service

This topic is an overview of the two options for tracking session information between HTTP requests in PingFederate.

The PingFederate server tracks user-session state information between HTTP requests, such as when PingFederate, acting as an identity provider (IdP), redirects a user’s browser to another system for authentication. When the user’s browser returns to PingFederate after authentication, the server needs access to the state associated with that user from before the redirect. Generally, this state is short-lived.

The `InterRequestStateMgmtProxy` implementation chooses between two methods to track this state: group RPC-based (the clustering default) and local memory-based (the standalone default).

The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-inter-request-state.conf` .

Group RPC-based session tracking

The group RPC-based implementation supports both adaptive clustering and directed clustering.

For adaptive clustering, PingFederate shares user session-state information with a replica set. If region identifiers are defined, PingFederate shares user session-state information among multiple replica sets across regions. You can override this default behavior in the `<pf_install>/pingfederate/server/default/conf/cluster-adaptive.conf` file.

For directed clustering, all preferred-node approaches are possible with this implementation.

The service point `InterRequestStateMgmt` in the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file uses the proxy `InterRequestStateMgmtProxy` to assign this implementation as the clustering default. The specific class name is `org.sourceid.saml20.service.impl.grouprpc.InterRequestStateMgmtGroupRpcImpl`

Local memory-based session tracking

The local memory-based session tracking implementation tracks users in the inter-request state in the local memory of the processing server. This is the standalone default.

> **Important**
>
> Adaptive clustering does not support this implementation. Use the group RPC-based session tracking instead.

The service point `InterRequestStateMgmt` in the `service-points.conf` file uses the proxy `InterRequestStateMgmtProxy` to assign this implementation as the clustering default. The specific class name is `org.sourceid.saml20.service.impl.localmemory.InterReqStateMgmtMapImpl`

Local memory-based session tracking and clustering

Group RPC-based session tracking is the clustering default. To use local memory-based session tracking in a clustered environment, update the service point `InterRequestStateMgmt` to use the local memory-based session tracking class, `org.sourceid.saml20.service.impl.localmemory.InterReqStateMgmtMapImpl`

> **Note**
>
> The load balancer must support sticky sessions to force all requests for the same user session to be routed to the same server.

Related links

- Adaptive clustering
- Directed clustering

IdP Session Registry Service

PingFederate uses the IdP Session Registry Service to facilitate single logout (SLO) by tracking assertions issued to Service Provider (SP) partners.

PingFederate uses this service only when acting in an Identity Provider (IdP) role and supports SLO with one or more partner connections.

When PingFederate is in clustered mode, the service proxy uses a group RPC-based, preferred-nodes implementation. The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-idp-session-registry.conf`.

This service supports both adaptive clustering and directed clustering.

For adaptive clustering, PingFederate shares user session-state information with a replica set. If region identifiers are defined, PingFederate shares user session-state information among multiple replica sets across regions. You can optionally override this default behavior in the configuration file.

For directed clustering, all preferred-node approaches are possible with this implementation.

> **Note**
>
> Both adaptive clustering and the subcluster deployment strategies in directed clustering do not support the SAML 2.0 SLO profile using the SOAP binding. If one or more SAML 2.0 connections are configured to support SLO via SOAP, you must choose between the sharing all nodes and designating state servers deployment strategies in directed clustering (see Directed clustering).

The service proxy uses the class `org.sourceid.saml20.service.impl.grouprpc.IdpSessionRegistryGroupRpcImpl`

> **Note**
>
> If the IdP session registry is configured with the Directed Clustering - Subclusters state management architecture, the capability to revoke sessions after password change or reset is not supported.

SP Session Registry Service

PingFederate uses the SP Session Registry Service to facilitate SLO by tracking assertions issued from identity provider (IdP) partners. PingFederate uses this service only when the server is acting in a service provider (SP) role and supports SLO with one or more partner connections.

When PingFederate is in clustered mode, the service proxy uses a group RPC-based, preferred-nodes implementation. The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-sp-session-registry.conf`.

This service supports both adaptive clustering and directed clustering.

For adaptive clustering, PingFederate shares user session-state information with a replica set. If region identifiers are defined, PingFederate shares user session-state information among multiple replica sets across regions. You can override this default behavior in the configuration file.

For directed clustering, all preferred-node approaches are possible with this implementation.

> **Note**
>
> Both adaptive clustering and the subcluster deployment strategies in directed clustering do not support the SAML 2.0 SLO profile using the SOAP binding. If one or more SAML 2.0 connections are configured to support SLO via SOAP, you must choose between the sharing all nodes and designating state servers deployment strategies in directed clustering (see Directed clustering).

The service proxy uses the class `org.sourceid.saml20.service.impl.grouprpc.SpSessionRegistryGroupRpcImpl`.

LRU memory management schemes

PingFederate uses an LRU memory manager to reduce memory usage by orphaned data.

During the normal course of transaction processing, the Inter-Request State-Management Service, the IdP Session Registry Service, and the SP Session Registry Service manage memory for PingFederate. End users abandoning web sessions results in orphaned data. To ensure that this orphaned data does not result in excessive memory usage, the data structures used by these services employ a least-recently-used (LRU) algorithm to purge old data. This algorithm automatically removes the oldest entries when a data structure reaches the maximum size.

Configure the maximum size of each data structure in the `<pf_install>/pingfederate/server/default/conf/size-limits.conf` file.

Assertion Replay Prevention Service

The Assertion Replay Prevention Service tracks POST assertions to prevent replay.

SAML standards specify that when a service provider (SP) receives assertions from the POST binding, the SP should keep track of each assertion for the duration of its validity to ensure that it is not replayed (that is, intercepted by a third party and re-posted). For OAuth and OpenID Connect, PingFederate can mandate a unique signed JSON Web Token (JWT) from the client for each request when the client is configured to authenticate with the `private_key_jwt` or `client_secret_jwt` client authentication method, to transmit request parameters using in signed request objects, or to do both. PingFederate delegates these responsibilities to the Assertion Replay Prevention Service.

When PingFederate is in clustered mode, the service proxy uses a group RPC-based, preferred-nodes implementation. The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-assertion-replay-prevention.conf`.

The Assertion Replay Prevention Service supports both adaptive clustering and directed clustering.

For adaptive clustering, PingFederate shares token (assertion or JWT) information with a replica set. If region identifiers are defined, PingFederate shares token information among multiple replica sets across regions. You can optionally override this default behavior in the configuration file for adaptive clustering.

For directed clustering, you must choose between the sharing all nodes and designating state servers deployment strategies in directed clustering for this service.

The service proxy uses the class `org.sourceid.saml20.service.impl.grouprpc.AssertionReplayPreventionServiceGroupRpcImpl`.

Unlike other services, the Assertion Replay Prevention Service fulfills only a security condition, rather than supporting normal SSO functionality, because there might be situations where the priority placed on cluster performance outweighs the priority placed on this security check. If you are in this situation, you have the option to change the implementation for the service point `AssertionReplayPreventionService` in the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file to one of these classes:

- `org.sourceid.saml20.service.impl.localmemory.AssertionReplayPreventionSvcInMemoryImpl`

  This is the implementation used in standalone mode. It performs all the appropriate replay checks but does not share any data with other nodes. A replay attempt routed to the same server node would fail, but other nodes would not have sufficient information to stop the transaction.

- `org.sourceid.saml20.service.impl.localmemory.AssertionReplayPreventionServiceNullImpl`

This implementation disables assertion-replay prevention. Use with caution when performance is an absolute priority.

Related links

- Adaptive clustering
- Directed clustering

Artifact-Message Persistence and Retrieval Service

PingFederate's Artifact-Message Persistence and Retrieval Service keeps track of one-time keys and associated data compliant with SAML and OAuth 2.0 standards.

The following standards require PingFederate to relay data to partners using a reference-style data transportation model and to guarantee that the reference keys are valid for one-time use only.

SAML artifact binding

PingFederate sends an artifact to the partner when transmitting SAML-outbound messages using the artifact binding. Later, the partner returns to PingFederate to exchange the artifact for the actual message. If the request is valid, PingFederate delivers the message and invalidates the artifact.

OAuth 2.0 authorization grant type

When processing an authorization request from an OAuth client that uses the authorization code grant type, PingFederate returns a code to the client based on specification. The client then includes that code in its token request to PingFederate to obtain an access token. If the request is valid, PingFederate delivers the access token and invalidates the code.

The Reference ID Adapter from the Agentless Integration Kit also applies the same data transportation model and one-time-use restriction in its drop-off and pick-up operations.

In a standard environment, the PingFederate server saves the data in memory, generates a key for the data, and sends the key to the partner. The Artifact-Message Persistence and Retrieval Service keeps track of the key and the associated data until the partner contacts the PingFederate server to exchange the key for the data.

Group RPC-based retrieval

When multiple PingFederate servers are deployed to form a cluster, the keys and their data are saved in the server that creates them. Because they are not replicated to other PingFederate servers, it is possible for a key resolution request to arrive at a server that does not contain the requested data. To handle this scenario, the Artifact-Message Persistence and Retrieval Service uses a group Remote Procedure Call (RPC) retrieval approach, where the server handling the key resolution request determines the data-hosting server based on the key value and contacts the appropriate server to retrieve the requested data. This group RPC implementation is compatible with the SAML artifact binding, the OAuth 2.0 authorization code grant type, and the Reference ID Adapter.

> **Note**
>
> The Artifact-Message Persistence and Retrieval Service also supports a local memory approach for SAML 2.0. This approach is only suitable in clustered environments where the OAuth 2.0 authorization code grant type and the Reference ID Adapter are not in use.

When PingFederate is in clustered mode, the service proxy selects a group RPC-based implementation, which takes advantage of node indexing but not the preferred-nodes concept. Sticky-session load-balancing strategies are not effective when the key request and its subsequent key resolution request can come from different locations.

Although this implementation does not take advantage of adaptive clustering or the preferred-nodes concept, you can configure the RPC time-out in the `<pf_install>/pingfederate/server/default/conf/cluster-artifact.conf` file.

SAML 2.0 indexing (local memory)

A SAML 2.0 federation entitycan support multiple artifact resolution services, each identified by a unique index number. Artifacts include this index, and a federation partner must send the artifact resolution request to the appropriate endpoint for that index. This means that servers do not need to share information concerning the artifact.

With this approach, partners must know about each of your backend servers. Generally, this means providing partners with a list that includes multiple artifact-resolution service endpoints with the corresponding indices.

> **Note**
>
> PingFederate does not automatically generate this information; an administrator must create it and send it to partners who are using the artifact binding.

For example, if you have four servers in a cluster, the list might look like this:

```
<ArtifactResolutionService Binding="...” Location="https://node1/idp/ARS.ssaml2" index="1"/>
<ArtifactResolutionService Binding="...” Location="https://node2/idp/ARS.ssaml2" index="2"/>
<ArtifactResolutionService Binding="...” Location="https://node3/idp/ARS.ssaml2" index="3"/>
<ArtifactResolutionService Binding="...” Location="https://node4/idp/ARS.ssaml2" index="4"/>
```

In this case, the index corresponds to the node index configured in the `run.properties` file on each individual server. This service encodes the node index in the artifact handle when running in a clustered mode (it will always use an index of `0` in standalone mode).

Partners also need direct access to each ARS endpoint, which can complicate your configuration of load balancers, proxies, and firewalls. This approach cannot be used for SAML 1.x, or with adapters that utilize PingFederate's artifact-data management.

To use this approach for SAML 2.0 federation deployments, edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file and change the implementation for the `artifact.store` service point to the class name `org.sourceid.saml20.service.impl.localmemory.ArtifactPersistenceServiceMapImpl`.

Back-Channel Session Revocation Service

PingFederate uses the Back-Channel Session Revocation Service to provide OAuth clients the capabilities to add sessions to the revocation list and to query the revocation status.

When PingFederate is in clustered mode, the service proxy uses a group remote procedure call (RPC)-based implementation. When adding a session to its revocation list, the processing node always propagates the information to all engine nodes in the cluster. This allows you to choose whether queries are processed locally or after collecting information from other engine nodes.

Processing queries locally results in faster response times for engine nodes in well-connected networks. Requiring data from other engine nodes adds a layer of protection against inconsistency among engine nodes revocation lists due to network outages.

You can configure the RPC timeout and other settings in the `<pf_install>/pingfederate/server/default/conf/cluster-session-revocation.conf` file.

The service proxy uses the class `org.sourceid.saml20.service.impl.grouprpc.SessionRevocationServiceGroupRpcImpl`.

FIFO memory management scheme

To ensure the revocation list does not result in excessive memory usage, the Back-Channel Session Revocation Service employs a First-In, First-Out (FIFO) algorithm to purge old data. When the maximum size is reached, the oldest entries are automatically removed.

The maximum number of sessions is configurable by the `SessionRevocationServiceMapImpl.max.revoked.sris` setting in the `<pf_install>/pingfederate/server/default/conf/size-limits.conf` file. The default value is `50000`.

The FIFO memory manager operates in addition to the **Session Revocation Lifetime** setting, which is globally configured in the **Authentication** > **Sessions** menu.

Related links

- Back-Channel Session Revocation
- Session Revocation API endpoint
- Runtime state-management architectures
- Configuring authorization server settings

Account Locking Service

The PingFederate Account Locking Service includes account lockout prevention and password spraying prevention.

Account lockout protection prevents user accounts from locking at the underlying user repository based on too many failed authentication attempts. It also adds a layer of protection against brute force and dictionary attacks because the user is locked out for a time period when the number of failed attempts exceeds the threshold. This protection is enabled in many areas of PingFederate, including the HTML Form Adapter, the Username Token Processor, the OAuth resource owner password credentials grant type, and the native authentication scheme for the administrative console and API.

Password spraying prevention adds a layer of defense against the attack pattern where bad actors try to gain access to protected resources by using the same password, typically weak or compromised, against multiple accounts from multiple locations. When enabled, PingFederate tracks the number of failed login attempts per password. When the number of failures for a particular password reaches a threshold, that password is temporarily locked out. Password spraying prevention applies to the HTML Form Adapter, the Username Token Processor, and the OAuth 2.0 resource owner password credentials grant type.

When PingFederate is in clustered mode, the service proxy uses a group remote procedure call (RPC)-based implementation. The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-account-locking.conf`.

This service supports both the adaptive clustering and directed clustering.

For adaptive clustering, PingFederate shares state information with a replica set. If region identifiers are defined, PingFederate shares state information among multiple replica sets across regions. You can override this default behavior in the `<pf_install>/pingfederate/server/default/conf/cluster-adaptive.conf` file.

For directed clustering, PingFederate shares state information across all nodes, which helps in scenarios where PingFederate is deployed behind a load balancing infrastructure without sticky sessions.

Related links

- Account lockout protection
- Password spraying prevention
- Adaptive clustering
- Directed clustering

Other services

PingFederate offers an account linking service and a pseudonym service to support SAML 2.0 federation deployment needs.

Account linking service

The account linking service stores the association between the external and internal identifiers of an end user when your implementation uses account linking as a service provider (SP) identity-mapping strategy. The default, standalone implementation uses a Java Database Connectivity (JDBC) interface to an embedded database within PingFederate. No information from the embedded database is shared across the cluster. When an identity provider (IdP) connection deployed in a cluster uses account linking, the default implementation will not work properly. In such cases, you must adjust the pointer for cluster use by pointing the service to an external database. For more information, see Define an account-linking data store.

Pseudonym service

The pseudonym service references the method needed by PingFederate to generate or look up a pseudonym for a user. PingFederate uses this service only if your site is acting in an IdP role and produces assertions containing pseudonyms as subject identifiers. The default implementation uses a message digest to produce the value so that no session-state synchronization is required. Developers who want to implement pseudonym handling differently can refer to the Javadoc reference describing `PseudonymService` interface for more information.

Deploying cluster servers

A PingFederate cluster consists of multiple nodes, each of which are likely running on a dedicated host system.

About this task

In a cluster, there are two types of nodes: engine nodes and administrative console nodes. Engine nodes service end user traffic, and multiple nodes are recommended to ensure high availability for your deployment. Only one administrative console node can be active in a given cluster. This node provides the user interface and administrative API that you can use to configure the cluster. Additionally, the administrative console node manages the following runtime functions:

- Performing periodic configuration archive backup
- Cleaning expired persistent authentication sessions
- Cleaning expired access grants
- Updating connections from metadata URLs, including PingOne SP connections if configured, and sending email notifications
- Performing automatic rotation of signing certificates if enabled

> **Note**
>
> Additional steps are required to set up failover for provisioning. If you are grouping servers exclusively to provide for provisioning failover, skip these steps and refer to Deploy provisioning failover.

These steps describe how to configure and deploy clustered PingFederate servers by editing each node in the `<pf_install>/pingfederate/bin/run.properties`.

Steps

1. Install PingFederate on each server in a cluster.
2. Edit the clustering properties of each node in the `<pf_install>/pingfederate/bin/run.properties` file. The following table provides information about each property:

Property	Description
`pf.operational.mode`	Controls the operational mode of the PingFederate server. PingFederate supports the following modes: **`STANDALONE`** ***(default)*** This server is a standalone instance that operates as the administrative console and runtime engine. **Important** The value `STANDALONE` should only be used in a cluster where session-state management is not needed for any reason and configuration-archive deployment is used as the configuration synchronization method. **`CLUSTERED_CONSOLE`** This server is part of a cluster and operates only the administrative console. **Important** By default, only one node in a cluster can run the administrative console. If the active/passive admin nodes feature is enabled, you can have more than one `CLUSTERED_CONSOLE`. **`CLUSTERED_ENGINE`** This server is part of a cluster and operates only as the runtime engine.

Property	Description
`pf.cluster.node.index`	Defines a unique index number for the server in a cluster. The index number is used to identify peers and optimize inter-node communication. The allowed range is `0` - `65535`. If no value is set for the node index, the system assigns an auto-generated value in the range of `0` to `2147483647`. This property has no default value. If you specify an index number, you can configure instances of the Cluster Node Authentication Selector and place them in authentication policies to customize authentication requirements based on the runtime node servicing a request.
`pf.cluster.auth.pwd`	Sets the password that each node in the cluster must use to authenticate when joining the cluster. This prevents unauthorized nodes from joining a cluster. The value can be any string or blank. **Note** Consider using a randomly-generated password with 22 or more alphanumeric characters. A strong, obfuscated, Jgroups cluster password can be generated with the **`clusterkey`** utility (`clusterkey.bat` for Windows and `clusterkey.sh` for Linux), located in the `<pf_install>/pingfederate/bin` directory. All nodes in a cluster must share the same value, blank or otherwise.
`pf.cluster.encrypt`	Indicates whether to encrypt network traffic sent between nodes in a cluster. The possible values are `true` or `false` (default). When set to `true`, communication within the cluster is encrypted with a symmetric key derived from the value of the `pf.cluster.auth.pwd` property. **Important** When the `pf.cluster.encrypt` property is set to `true`, you must provide a value for the `pf.cluster.auth.pwd` property. Otherwise PingFederate aborts during its startup process. All nodes in a cluster must have the same value for this property.
`pf.cluster.encryption.keysize`	The length of the key that PingFederate takes into consideration when deriving the symmetric key from the value of the `pf.cluster.auth.pwd` property for the purpose of encrypting network traffic sent between nodes in a cluster. Required only when the `pf.cluster.encrypt` is set to `true`. All nodes in a cluster must have the same value set for this property. The default value is `128`.

Property	Description
`pf.cluster.bind.address`	Defaults to `NON_LOOPBACK`, which leaves the system to choose an available non-loopback IP address. Alternatively, enter an IP address of the network interface to which the cluster communication should bind. For machines with more than one network interface, provide a specific IP address. You can use this property to increase performance (particularly with UDP) and improve security by segmenting cluster-communication traffic onto a private network or VLAN. **Tip** Besides `NON_LOOPBACK` or an IP address, you can also use other values supported by JGroups. For more information, see the `bind_addr` special values in .org/manual/index.html//[JGroups documentation]. **Important** This field does not support DNS name. Use the default value *NON_LOOPBACK* or replace it with an IP address.
`pf.cluster.bind.port`	Specifies the port associated with the `pf.cluster.bind.address` property or with the default network interface used. This is the port used by other cluster members during their discovery process, usually via the `pf.cluster.tcp.discovery.initial.hosts` property. The default value is `7600`.
`pf.cluster.failure.detection.bind.port`	Indicates the bind port of a server socket that is opened on the given node and used by other nodes as part of the cluster's failure-detection mechanisms. If set to `0` or unspecified, a random available port is used. The default value is `7700`.

Property	Description
`pf.cluster.transport.protocol`	Indicates the transport protocol used for cluster communication. Values are `udp` or `tcp`. The default value is *tcp*. All nodes in a cluster must have the same value set for this property. Use UDP when IP multicasting is enabled in the network environment and the majority of cluster traffic is point-to-full-group. You must also configure both the `pf.cluster.mcast.group.address` and `pf.cluster.mcast.group.port` properties. Use TCP for geographically dispersed servers or when multicast is not available or disabled for some other reason. For example, when using routers that do not support multicast messaging. TCP might also be appropriate if your cluster configuration employs more point-to-point or point-to-few messaging than point-to-group.You must also configure the `pf.cluster.tcp.discovery.inital.hosts` property. ⓘ **Note** This property is a reference to a protocol-stack XML configuration file located in the `<pf_install>/pingfederate/server/default/conf/` directory. Two stacks are provided: one for UDP multicast and one for TCP. You can customize either stack or add to it as needed by modifying the associated configuration file.
`pf.cluster.mcast.group.address`	Defines the IP address shared among nodes in the same cluster for UDP multicast communication; required when UDP is set as the transport protocol. The valid range is `224.0.0.0` - `239.255.255.255`. Some addresses in this range are reserved for other purposes. This property is not used for TCP. All nodes in a cluster must have the same value set for this property. The default value is *239.16.96.69*.
`pf.cluster.mcast.group.port`	Defines the port in conjunction with the `pf.cluster.mcast.group.address` property value. This property is not used for TCP configurations. All nodes in a cluster must have the same value set for this property. The default value is `7601`.

Property	Description
`pf.cluster.tcp.discovery.initial.hosts`	Designates a static list of PingFederate servers to be contacted for cluster membership information when discovering, joining, and rejoining the cluster. This value is required when TCP is set as the transport protocol. The value is a comma-separated list of host names (or IP addresses) and their cluster bind ports, for example, `host1[7600],10.0.1.4[7600],host7[1033],10.0.9.45[2231]`. When using static discovery, add at least one node for the cluster to know in advance. This property should contain all nodes in the cluster (including itself) to increase the likelihood of new members finding and joining the cluster. When using dynamic discovery, leave this property blank and enable dynamic discovery in the `<pf_install>/pingfederate/server/default/conf/tcp.xml` file. Learn more in Enabling dynamic discovery for clustering.
`pf.cluster.adaptive`	Indicates whether runtime state-management services should use the adaptive clustering architecture. The default value is `true` for new installations and `false` for upgrades.
`pf.cluster.diagnostics.enabled`	`false` turns off JGroups diagnostics. `true` turns it on. The default value is `false`.
`pf.cluster.diagnostics.addr` and `pf.cluster.diagnostics.port`	The multicast address and port this node listens on for diagnostic messages. The default values are `224.0.75.75` and `7500`, respectively. Do not change the default values.

`node.tags`	Defines the tags associated with this node.
	Configuration is optional. When configured, PingFederate considers this property when processing requests. For example, you can use tags to determine the datastore location that this PingFederate node communicates with. You can also use tags in conjunction with authentication selectors and policies to define authentication requirements.
	Node tags only apply to engine nodes (CLUSTERED_ENGINE). If you configure a node tag for the admin node (CLUSTERED_ADMIN), it will not appear in the **Cluster Management** overview.
	You can specify one tag.
	`node.tags=north`
	You can also specify a list of prioritized, space-separated tags.
	`node.tags=1 123 234`
	Tags cannot contain spaces.

3. **Optional:** Edit configuration files in each node that control the cluster protocol and runtime state-management service. For more information, see Runtime state-management architectures and Runtime state-management services.
4. **Optional:** If outbound provisioning is configured for your site and you want to provide failover capabilities, identify and configure the provisioning failover nodes. For more information, see Deploy provisioning failover.
5. Start or restart PingFederate on all nodes.
6. Sign on to the administrative console.
7. If you have not done so, import your PingFederate license. Learn more in License management.
8. On the **System > Server > Cluster Management** page, click **Replicate Configuration** to push the license information from the console node to all engine nodes.

Result

After you set up the clustered environment, you can start configuring PingFederate through the administrative console. When PingFederate detects a change, it prompts you to replicate the configuration to all engine nodes.

Dynamic cluster discovery

Instead of configuring a static list of known PingFederate nodes in advance, dynamic cluster discovery lets you configure new nodes to pull cluster membership information from a centralized repository.

Dynamic discovery is well-suited for environments where traffic volume could spike and require additional resources during peak hours. Because safe storage and ready accessibility of the information by all nodes is crucial, PingFederate supports identity and access management (IAM) roles for Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), and OpenStack Swift. The dynamic discovery method requires only a one-time setup.

For information about configuring dynamic cluster discovery, see Enabling dynamic discovery for clustering.

Dynamic cluster discovery protocols

In addition to the static cluster discovery protocol, `TCPPING`, PingFederate supports the following dynamic discovery protocols:

- `NATIVE_S3_PING`
- `DNS_PING`
- `AWS_PING`
- `SWIFT_PING`

> **Note**
>
> The `S3_PING` discovery method has been deprecated because of the Amazon Web Services (AWS) deprecation of the SigV2 signing method. When deployed in AWS, the recommended discovery method is `NATIVE_S3_PING`. See the JGroups documentation for alternatives when deployed in other environments.

`NATIVE_S3_PING` and `SWIFT_PING` enable the flexibility to use both public and private cloud storage. PingFederate maintains cluster membership information in a centralized repository, a bucket in Amazon Simple Storage Service (Amazon S3) or a container in an OpenStack infrastructure.

PingFederate contacts the repository for a list of nodes. If PingFederate receives at least one node, a cluster exists, and it joins the cluster and updates the repository with its information, including its IP address. If PingFederate receives no node, it forms a new cluster and updates the repository with its information so that the next node can find the new cluster. When PingFederate shuts down, it removes itself from the list and pushes an update to the repository.

`NATIVE_S3_PING` uses the AWS SDK and provides a stable connection by using built-in security features, such as obtaining credentials through IAM server instance profiles. This protocol is the recommended dynamic discovery mechanism when you're running in AWS but aren't using Kubernetes.

`DNS_PING` uses DNS `A` or `SRV` records to perform discovery. This protocol is the recommended dynamic discovery mechanism when using Kubernetes. For more information, see the JGroups documentation about the .jgroups.org/manual4/index.html// [DNS_PING] protocol.

`AWS_PING` lets you scale your PingFederate infrastructure using Amazon EC2 instances in the AWS cloud, in one or multiple regions. PingFederate queries AWS for a list of eligible EC2 instances. If PingFederate receives at least one node, a cluster exists, and it joins that cluster. If PingFederate receives no node, it forms a new cluster.

You must enable permissions to `ec2:Describe*` actions in the AWS IAM role assigned to the EC2 instance or associate them with the access_key parameter that you provide as part of the dynamic discovery configuration. You can also use a combination of tags and filters, in which case only EC2 instances that satisfy both criteria are returned.

Discovery mechanisms versus runtime state-management architectures

Discovery mechanisms are separate from runtime state-management architectures. Discovery mechanisms determine how to find nodes to retrieve cluster information for the purpose of joining and rejoining a cluster. Runtime state-management architectures determine which nodes session-state information is shared to and fetched from.

PingFederate supports adaptive clustering and directed clustering runtime state-management architectures. When opting for dynamic discovery, consider enabling adaptive clustering whenever possible. If multiple regions are involved, configure multi-region support for adaptive clustering as well. For more information and configuration steps, see Adaptive clustering.

Regardless of the chosen runtime state-management architecture, all nodes must still be able to communicate with other nodes for clustering-protocol messages. For more information, see Runtime state-management architectures.

Related links

- Deploying cluster servers

Enabling dynamic discovery for clustering

You can enable and configure dynamic discovery setup for clustered PingFederate deployments by editing the `run.properties` and `jgroups.properties` files.

Before you begin

If your PingFederate deployment has cluster discovery settings in the `tcp.xml` file, migrate those settings to the `jgroups.properties` file. Learn more in Migrating cluster discovery settings.

About this task

> **Important**
>
> If you enable dynamic discovery, you must manually configure or synchronize the dynamic discovery properties in the `run.properties` file and `jgroups.properties` file on each node in the cluster. The files are not synchronized automatically across the nodes in a cluster, nor is it part of the replicate configuration process.
> Whenever you add a node to the cluster, perform the following procedure on the new node.

To enable and configure dynamic discovery for clustering:

Steps

1. Configure the cluster protocol properties in the `<pf_install>/pingfederate/bin/run.properties` file.

   See the file's inline comments and the following table.

Property	Description
`pf.operational.mode`	Configure the operational mode of the PingFederate node. The value `CLUSTERED_CONSOLE` denotes an administrative console node. The value `CLUSTERED_ENGINE` denotes a runtime engine node. A PingFederate cluster has only one administrative console node. When you scale your PingFederate infrastructure, set the `pf.operational.mode` property value to `CLUSTERED_ENGINE` to deploy additional PingFederate runtime engine nodes.
`pf.cluster.tcp.discovery.initial.hosts`	Remove any configured value. No IP addresses are required here.
`pf.cluster.transport.protocol`	Set the value to `tcp`.
`pf.cluster.*`	See the inline comments to configure the rest of the `pf.cluster.*` property values.

2. Configure dynamic discovery properties in the `<pf_install>/pingfederate/bin/jgroups.properties` file.

   See the file's inline comments and the following tables.

Property	Description
`pf.cluster.NATIVE_S3_PING.region_name`	The name of the region in which discovery should be attempted. If no region is specified, only nodes in the same region as this node can be discovered.
`pf.cluster.NATIVE_S3_PING.bucket_name`	The name of the bucket in your Amazon S3 environment. You can find more information about buckets in the Create a bucket documentation from Amazon.

Property	Description
`pf.cluster.NATIVE_S3_PING.remove_all_data_on_view_change`	When set to `true` (the default), JGroups cleans up data when it detects a view change. Learn more in Reliable group communication with JGroups.
`pf.cluster.NATIVE_S3_PING.write_data_on_find`	Default value of `true` resolves the issue where subclusters could fail to merge after a network partition.

Property	Description
`pf.cluster.DNS_PING.dns_query`	A DNS query to the DNS Server that obtains information about the cluster members. For example, `jgroups-dns-ping.myproject.svc.cluster.local`. Learn more in Reliable group communication with JGroups.

Property	Description
`pf.cluster.AWS_PING.port_number`	The port on which PingFederate listens for cluster communication. The default value is `$\{pf.cluster.bind.port}`, which pulls the value defined by the `pf.cluster.bind.port` property in the `<pf_install>/pingfederate/bin/run.properties` file.
`pf.cluster.AWS_PING.port_range`	The number of additional ports that PingFederate can probe when attempting to connect to other nodes in the event that it fails to connect to the port specified by the `port_number` property. For example if the `port_number` property is `7600`, a `port_range` property value of `0` means PingFederate will only try to connect at port 7600. If the `port_range` property value is set to `2`, PingFederate will make up to two additional attempts, with the port value increasing by one each time. For example, if PingFederate fails to connect at port 7600, it will try at port 7601. If it fails again, it will try at port 7602. The default value is `0`.
`pf.cluster.AWS_PING.regions`	A comma separated list of EC2 regions in which PingFederate will attempt discovery. If no regions are specified, only nodes in the same region as this node can be discovered. List all regions where you have nodes running. You can find more information about regions in the AWS documentation on Regions, Availability Zones, and Local Zones.
`pf.cluster.AWS_PING.tags`	A comma separated list of EC2 tag names. When specified, only EC2 instances that have been assigned with the specified tags can be discovered. If multiple tags are specified, only EC2 instances that have been assigned with all tags can be discovered. You can find more information about tags in the EC2 resources documentation from Amazon.

Property	Description
`pf.cluster.AWS_PING.filters`	A semi-colon separated list of key value pairs of system metadata. When specified, only EC2 instances that match the specified filters can be discovered. If multiple filters are specified, only EC2 instances that match all filters can be discovered. Note that you can enter a comma separated list of values for each filter. In this case, a filter is considered a match if one of the values is satisfied. For example, if you enter: `filters="instance-type=t2.small,t2.medium;architecture=x86_64"` , then only EC2 instances that have an `instance-type` of either `t2.small` or `t2.medium` and also an `architecture` of `x86_64` can be discovered. You can find more information about filters in the describe-instances documentation from Amazon.
`pf.cluster.AWS_PING.access_key` and `pf.cluster.AWS_PING.secret_key`	The access key and its secret key for the purpose of querying AWS for EC2 instances. Applicable and required only if permissions to `ec2:Describe*` actions are associated with the access key. Keys can be encrypted using the **`obfuscate`** utility (`obfuscate.bat` for Windows or `obfuscate.sh` for Linux), located in the `<pf_install>/pingfederate/bin` directory
`pf.cluster.AWS_PING.log_aws_error_messages`	When set to `true` (the default), error messages received from AWS are logged to the server log.

Property	Description
`pf.cluster.SWIFT_PING.auth_type`	The authentication type.
`pf.cluster.SWIFT_PING.auth_url`	The authentication URL.
`pf.cluster.SWIFT_PING.username` and `pf.cluster.SWIFT_PING.password`	The security credentials. Password can be encrypted using the **`obfuscate`** utility (`obfuscate.bat` for Windows or `obfuscate.sh` for Linux), located in the `<pf_install>/pingfederate/bin` directory.
`pf.cluster.SWIFT_PING.tenant`	The name of your OpenStack Keystone tenant.
`pf.cluster.SWIFT_PING.container`	The name of the root container. You can find more information about each of the `SWIFT_PING` properties in the JGroups SWIFT_PING documentation.
`pf.cluster.SWIFT_PING.remove_all_data_on_view_change`	When set to `true` (the default), JGroups cleans up data when it detects a view change. Learn more in the JGroups FILE_PING documentation.

> **Note**
>
> To use a discovery protocol that PingFederate does not officially support, set the discovery protocol and class in `jgroups.properties`:
>
> ```
> pf.cluster.discovery.protocol={discovery protocol}
> pf.cluster.discovery.class={fully qualified class name}
> ```
>
> For example, to use a discovery protocol called CUSTOM, add the following lines:
>
> ```
> pf.cluster.discovery.protocol=CUSTOM
> pf.cluster.discovery.class=org.jgroups.custom.discovery
> ```
>
> To add a custom attribute to a protocol, use the following syntax in `jgroups.properties`:
>
> ```
> pf.cluster.{discovery protocol}.{attribute name}={attribute value}
> ```
>
> For example, to add a custom attribute called `bucket_prefix` to the NATIVE_S3_PING protocol and assign it the value `jgroups`, add the following line:
>
> ```
> pf.cluster.NATIVE_S3_PING.bucket_prefix=jgroups
> ```

3. Start or restart PingFederate.

4. Repeat these steps for each node in the cluster.

Related links

- Dynamic cluster discovery

Migrating cluster discovery settings

To simplify future upgrades, you should migrate the cluster discovery settings in the `tcp.xml` file to the `jgroups.properties` file.

About this task

When you upgrade, PingFederate can set the default configuration of any new features in the supported discovery protocols. Because of this, you should use the `<pf_install>/pingfederate/bin/jgroups.properties` file to enable and configure cluster discovery settings. For example, if `NATIVE_S3_PING` gets a new discovery protocol configuration, a PingFederate upgrade could automatically add the following line to `jgroups.properties` as a reasonable default, letting you take advantage of the new configuration:

```
pf.cluster.NATIVE_S3_PING.new_configuration=reasonable-default-configuration-value
```

For more information, see Dynamic cluster discovery.

Note

The `tcp.xml` file has a `${DISCOVERY_TAG}` that replaces all content related to cluster discovery protocol configuration. The `jgroups.properties` file is where the discovery protocol configuration exists and lets you do what you previously did in `tcp.xml`, such as:

- Add custom configuration to discovery protocols that PingFederate officially supports, using the following syntax:

  ```
  pf.cluster.{discovery protocol}.{attribute name}={attribute value}
  ```

  For example, the following line adds a custom attribute called `bucket_prefix` with the value `jgroups` to the `NATIVE_S3_PING` protocol configuration:

  ```
  pf.cluster.NATIVE_S3_PING.bucket_prefix=jgroups
  ```

- Add and configure a discovery protocol that PingFederate does not officially support, using the following syntax:

  ```
  pf.cluster.discovery.protocol={discovery protocol}
  pf.cluster.discovery.class={fully qualified class name}
  ```

  For example, the following lines add a discovery protocol called `CUSTOM` and then add an attribute called `attr1` with the value `value1` to it:

  ```
  pf.cluster.discovery.protocol=CUSTOM
  pf.cluster.discovery.class=org.jgroups.custom.discovery
  pf.cluster.CUSTOM.attr1=value1
  ```

The examples in the following table compare the syntax of cluster discovery settings in the `tcp.xml` file and the `jgroups.properties` file.

Examples intcp.xml	Examples injgroups.properties

`<TCPPING`
`  initial_hosts="$`
`{pf.cluster.tcp.discovery.initial.hosts}"`
`  return_entire_cache="true"`
`  port_range="0"/>`
	`pf.cluster.TCPPING.initial_hosts=$`
	`{pf.cluster.tcp.discovery.initial.hosts}`
	`pf.cluster.TCPPING.return_entire_cache=true`
	`pf.cluster.TCPPING.port_range=0`

`<org.jgroups.aws.s3.NATIVE_S3_PING`
`  region_name="us-east-1"`
`  bucket_name="[...]"`
`  endpoint="[...]"`
`  remove_all_data_on_view_change="true"`
`  write_data_on_find="true"/>`
	`pf.cluster.NATIVE_S3_PING.region_name=us-east-1`
	`pf.cluster.NATIVE_S3_PING.bucket_name=`
	`pf.cluster.NATIVE_S3_PING.endpoint=`
	`pf.cluster.NATIVE_S3_PING.remove_all_data_on_view_change=true`
	`pf.cluster.NATIVE_S3_PING.write_data_on_find=true`

Examples intcp.xml	Examples injgroups.properties

`<dns.DNS_PING`
`  dns_query="[service`
`name].default.svc.cluster.local" />`
	`pf.cluster.DNS_PING.dns_query=[service`
	`name].default.svc.cluster.local`

To migrate the cluster discovery settings from the `tcp.xml` file to the `jgroups.properties` file:

Steps

1. Open the `<pf_install>/pingfederate/bin/jgroups.properties` file and the `<pf_install>/pingfederate/ server/ default/conf/tcp.xml` file in a text editor.
2. In the `jgroups.properties` file:
    1. Specify the cluster discovery protocol.

        For example:

        ```
        pf.cluster.discovery.protocol=NATIVE_S3_PING
        ```

    2. Specify the values for the cluster discovery protocol parameters that correspond with the values in the `tcp.xml` file.

        For example:

        ```
        pf.cluster.NATIVE_S3_PING.region_name=us-east-1
        pf.cluster.NATIVE_S3_PING.bucket_name=
        pf.cluster.NATIVE_S3_PING.endpoint=
        pf.cluster.NATIVE_S3_PING.remove_all_data_on_view_change=true
        pf.cluster.NATIVE_S3_PING.write_data_on_find=true
        ```

3. In the `tcp.xml` file, insert the `${DISCOVERY_TAG}` and remove the cluster discovery protocol settings.

    > **Note**
    >
    > Do not remove other settings from the `tcp.xml` file.

4. Repeat these steps for the other PingFederate nodes in the cluster.

Deploying provisioning failover

After configuring outbound provisioning, you can set up one or more PingFederate failover servers specifically for provisioning backup.

About this task

Provisioning runtime processing and failover is independent of single sign-on (SSO) or single logout (SLO) runtime processing and server clustering. However, if you are already deploying, or have deployed, a cluster for federation-protocol runtime processing, you can use a subset of those servers for provisioning failover. Alternatively, you can mix the configuration or set up provisioning-failover servers independently.

> **Note**
>
> Each server in the failover network must be configured to use the same relational database.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Steps

1. Select two or more runtime instances of PingFederate to configure for provisioning failover.
2. For each server instance, edit provisioning properties in the `<pf_install>/pingfederate/bin/run.properties` file as follows:

Property	Description
`pf.provisioner.mode`	The status of outbound provisioning. Allowed values are: **`OFF`** ***(default)*** Outbound provisioning is disabled. **`STANDALONE`** Provisioning is enabled, without failover. **`FAILOVER`** Provisioning is enabled, with failover. **Important** The value `STANDALONE` cannot be used for failover configuration. This property must be set to `FAILOVER` on the primary and secondary servers.
`provisioner.node.id`	The unique index number of the provisioning server. Each server must have a unique index number, which is used to prioritize which server is currently active and which is next in line in case of a failure. Values are any number. **Important** The primary active primary server should have an index number of 1. The lowest value in the environment becomes the primary. These node IDs are not required to start at 1, but it is recommended that they start at 1. The node IDs for each node must increase sequentially. The number must not exceed the maximum integer value supported by Java, which is 2147483647. Initial start-up performance degradation may result if the node ID does not start at 1.

Property	Description
`provisioner.failover.grace.period`	The time interval (in seconds) between the first indication that a node is dead and failover to the next server in line. The time period should be greater than the **Synchronization Frequency** set in the **System > Server > Protocol Settings > Outbound Provisioning** tab on the administrative console. The default value is `600`, which is 10 minutes.

> **Important**
>
> You must seperately configure the failover properties in the `run.properties` file on each provisioning server, because the `run.properties` file is not copied among the provisioning servers automatically or as part of the **Replicate Configuration** process.

3. Start or restart all of the PingFederate servers.

4. If you have not already done so, set up an external database to facilitate provisioning and then update the **Provisioning Data Store** setting on the **System > Server > Protocol Settings > Outbound Provisioning** tab. See Configuring outbound provisioning settings for more information.

5. After configuration, if the provisioning servers belong to the same PingFederate clustered environment, go to the **System > Server**. In the **Cluster Management** window, replicate the new **Provisioning Data Store** setting to all nodes. If the provisioning servers are individual PingFederate servers, for each provisioning server, create a datastore connection to the same external database and update the **Provisioning Data Store** setting manually.

Configuration synchronization

All nodes in a PingFederate clustered environment must have the same configuration settings, as set through the administrative console. You can use any of the following methods to ensure that configuration data is synchronized on all cluster nodes.

- Push from the administrative console.
- Deploy configuration archive.
- Make a RESTful API call to the `/cluster` administrative API endpoint.
- Make a web service call to the `/pf-mgmt-ws/ws/ConfigReplication` Connection Management Service endpoint.

> **Note**
>
> Changes made directly to configuration files must be replicated manually across the cluster, as applicable. If the PingFederate servers are running, you must restart them after you replicate the changes.

Console configuration push

When multiple PingFederate servers are set up to run as a cluster, the administrative console provides a **Cluster Management** window.

Whenever applicable changes are made through the administrative console, a message appears at top of the console as a reminder to go to the **Cluster Management** window and to replicate the current console configuration to all server nodes in the cluster.

> **Note**
>
> You must also use the **Replicate Configuration** window to initiate the transmission of the license file from the console node to all server nodes.

The **Cluster Management** window is also useful for verifying the current member servers of your PingFederate cluster.

Configuration-archive deployment

Uploading configuration archives is an alternate method of copying configurations to clustered PingFederate servers.

After you configure or reconfigure the console, you can also update cluster nodes by downloading a configuration archive from the **System > Server > Configuration Archive** window and then deploying it either manually or using a scripted process to the `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory on each cluster node or provisioning-failover server.

> **Note**
>
> If you use the drop-in deployment process:
>
> - PingFederate will not let you import the configuration archive of an older or newer version, and to ensure successful importation of the configuration archive file with this process, you must rename the file `data.zip` .
> - On startup, the heartbeat endpoint will not return `200` until the archive import completes. If you have configured a health check or probe that can trigger a restart of the server, crash loop behavior can result. Review the configuration of these checks to ensure time thresholds are set appropriately.

A configuration archive contains the same information sent during the configuration push from the administrative console described in Console configuration push. However, configuration-archive deployment also provides for scheduling and scripting cluster synchronization.

Runtime state-management services

If you have configured one of the following runtime state-management services on the engine nodes, you must manually migrate the configuration files to the engine nodes. The configuration files are locted at `<pf_install>/pingfederate/server/default/conf`

Configuration file and service implementation

Configuration file	RPC-based service implementation
`cluster-account-locking.conf`	Account Locking Service
`cluster-artifact.conf`	Artifact-Message Persistence and Retrieval Service

Configuration file	RPC-based service implementation
`cluster-assertion-replay-prevention.conf`	Assertion Replay Prevention Service
`cluster-idp-session-registry.conf`	IdP Session Registry Service
`cluster-inter-request-state.conf`	Inter-Request State-Management (IRSM) Service
`cluster-session-revocation.conf`	Back-Channel Session Revocation Service
`cluster-sp-session-registry.conf`	SP Session Registry Service

Administrator’s Reference Guide

This guide provides information about using PingFederate to deploy a secure internet single sign-on (SSO) solution based on the latest security and e-business standards.

Use this guide to learn about the following:

- Attribute mapping expressions
- Authentication policies
- Bundled adapters
- Customer IAM configuration
- Customizing assertions and authentication requests
- Fulfillment by datastore queries
- IdP-to-SP bridging
- Identity provider SSO configuration
- OAuth configuration
- Security management
- Self-service user account management
- Service provider SSO configuration
- System administration
- System settings
- Troubleshooting
- WS-Trust STS configuration

Attribute mapping expressions

As of PingFederate 10.1, the use of expressions is enabled by default. You can use the Expression Admin administrative role to map user attributes by using OGNL expressions.

If you upgraded to PingFederate 10.1 from a previous version, the use of expressions is still enabled or disabled based on the configuration in the earlier version. Also, when upgrading PingFederate to 10.1 or later, administrative users who were granted the Admin role in the earlier installation are granted the Expression Admin role automatically.

Enabling and disabling expressions

As of PingFederate 10.1, the use of expressions is enabled by default. You can manually disable the use of expressions by editing a configuration file.

About this task

When upgrading PingFederate to 10.1 or later, administrative users who were granted the Admin role in the earlier installation are granted the Expression Admin role automatically.

You can disable the use of expressions by setting `evaluateExpressions` to false as described in the following procedure. Also, go to **System > Server > Administrative Accounts** and remove the **Expression Admin** role from all Admin users. Doing this will prevent Admin users from entering expressions into PingFederate if the `evaluateExpressions` element is set to true at a later time. For more information, see Administrative accounts.

> **Important**
>
> If the current configuration contains expressions, disabling the feature causes errors during runtime processing.

Steps

1. Edit the `org.sourceid.common.ExpressionManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   > **Note**
   >
   > If you have a clustered PingFederate environment, edit the configuration file on the console node.

2. Change the value of the element named `evaluateExpressions` to either `true` or `false` and save the file.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <config xmlns="http://www.sourceid.org/2004/05/config">
       <item name="evaluateExpressions">true</item>
   </config>
   ```

   > **Note**
   >
   > The absence of an installed default value does not necessarily disable the use of expressions. You can successfully import configuration archives containing expressions to facilitate backward compatibility when no value is present, and further use of the feature is enabled. The term "silent" is used for this condition in the server log.

3. If you have a stand-alone PingFederate environment, start or restart PingFederate.

   > **Tip**
   >
   > If you are enabling expressions to use for mapping outbound provisioning attributes, you do not need to restart the PingFederate server.

4. If you have a clustered PingFederate environment:

   1. Sign on to the PingFederate administrative console.
   2. From **System > Server > Cluster Management**, click **Replicate Configuration**.

Result

When you enable expressions, they are available for use in multiple locations:

- The **Source** list under each of the administrative-console contract fulfillment windows

- The **Show Advanced Criteria** section on the **Issuance Criteria** window following each of the administrative-console contract fulfillment windows
- The provisioning attribute-mapping window when the **Outbound Provisioning** protocol is enabled

Construct OGNL expressions

Use OGNL expressions and syntax to evaluate and manipulate attribute values and return information based on the results.

OGNL is based on the Java programming language. You can transform a range of values into a text description or do the same for a sequence of ranges.

Use the `#` symbol to reference OGNL variables. For an identity provider (IdP), PingFederate provides predefined OGNL variables for IdP-adapter attributes, any attributes retrieved from datastores, and attributes for token authorization. For a service provider (SP), variables are available for attributes received in an assertion, an attribute query, and attributes for token authorization. For example, you can retrieve the `SAML_SUBJECT` value with `#SAML_SUBJECT`.

> **Note**
>
> Use the following construction for any attributes from any source that contain special characters that cannot be parsed by OGNL: `#this.get("<attribute_name>")`.

> **Note**
>
> Because OGNL uses the "at" symbol (`@`) to reference static Java methods, expressions containing the symbol must be enclosed in double quotes. Otherwise, expression parsing fails. For example, use `#SAML_SUBJECT="usr@msn.com"`, not `#SAML_SUBJECT=usr@msn.com`.

Data store syntax

For datastore attributes with an attribute source ID, use the `#this.get("ds.attr-source-id.attribute_name")` syntax.

For datastore attributes without an attribute source ID, use the `#this.get("ds.attribute_name")` syntax.

Other variable syntax

To access mapped attributes, use the `#this.get("mapped.attribute_name")` syntax.

To access most context attributes, use the `#this.get("context.attribute_name")` syntax.

To access the HTTP Request context attribute, use the `#this.get("context.HttpRequest").getObjectValue()` syntax.

> **Note**
>
> The returned value is an instance of `javax.servlet.http.HttpServletRequest`. See http://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletRequest.html.

Related links

- The OGNL language guide from Apache Commons

Sample OGNL expressions

OGNL expressions provide the flexibility to evaluate and manipulate values. These applications include using the following expressions to determine net worth, form a single sign-on (SSO) token, verify a user's group, retrieve a value from an HTTP request object, and check the authenticity of a client certificate.

General

In this sample expression, the value of the attribute "net-worth" is transformed first to eliminate any dollar signs or commas, then the result is evaluated to determine whether the user's net worth falls into a "bronze," "silver," or "gold" category.

```
#result=#this.get("net-worth").toString(),
#result=#result.replace("$",""),
#result=#result.replace(",",""),
#result < 500000 ? "bronze" :
#result < 1000000 ? "silver" : "gold"
```

Multivalued attribute

```
new org.sourceid.saml20.adapter.attribute.AttributeValue( {"Blue", "Gray", "Pink"})
```

This expression formulates a multivalued attribute in an SSO token.

```
<saml:Attribute Name="clrs" ...>
  <saml:AttributeValue ...>Blue</saml:AttributeValue>
  <saml:AttributeValue ...>Gray</saml:AttributeValue>
  <saml:AttributeValue ...>Pink</saml:AttributeValue>
</saml:Attribute>
```

and

```
 {
   ...,
   "clrs": [
     "Blue",
     "Gray",
     "Pink"
   ],
   ...
 }
```

In these truncated samples, `clrs` is the multivalued attribute. The former is a SAML assertion through a SAML service provider (SP) connection. The latter is a JSON web token (JWT) through a WS-Federation SP connection using JWT as the token type.

Token authorization

This expression verifies whether a user is a member of the "Engineering" or "Marketing" group.

```
#this.get("ds.memberOf")!=null?
(
  (
    #this.get("ds.memberOf").hasValue("CN=Eng,OU=E,DC=contoso,DC=com")
    &&
    #this.get("context.VirtualServerId").toString().equals("Engineering")
  )
  ||
  (
    #this.get("ds.memberOf").hasValue("CN=Mkt,OU=M,DC=contoso,DC=com")
    &&
    #this.get("context.VirtualServerId").toString().equals("Marketing")
  )
):false
```

The following expression extracts the domain information out of an email address (`mail`) and returns true if it matches a specific domain.

```
#this.get("mail")!=null?
(
  #email=#this.get("mail").toString(),
  #atSign="@",
  #at=#mail.indexOf(#atSign),
  #at > 0?
    (
      #domain=#mail.subject(#at+1),
      #domain.matches("(?i)example.com")
    ):false
):false
```

Note

Line breaks are inserted to both samples for readability only. You must enter statements calling methods whose arguments are enclosed in quote on a single line.

This sample expression returns true when the IP address of the client is within the specified CIDR range of `fe80::74da:14b:76d1:eba3/128`.

```
#isWithinCidrRange =
@com.pingidentity.sdk.CIDROperations@isInRange(#this.get("context.ClientIp"),"fe80::74da:14b:
76d1:eba3/128")
```

The `isInRange` method supports both IPv4 and IPv6 CIDR notations.

HTTP request context

You can use the following example to retrieve a value from an HTTP request object. The expression retrieves the `User-Agent` HTTP header value and compares it against a value required for token authorization.

```
#this.get("context.HttpRequest").getObjectValue().getHeader("User-Agent").equals("somevalue")
```

STS client authentication context

This security token service (STS) SSL Client Certificate Chain example checks that the issuer of the client certificate matches the specified distinguished name (DN).

```
#this.get("context.StsSSLClientCertChain").getObjectValue()[1].getSubjectX500Principal().equals(new
javax.security.auth.x500.X500Principal("CN=Ping Identity Engineering,OU=Engineering,O=Ping
Identity,L=Denver,ST=CO,C=USA"))
```

> **Note**
>
> `#this.get("context.StsSSLClientCertChain").getObjectValue()` returns an array of `java.security.cert.X509Certificate` instances. This array starts with the client certificate itself.
> For more information, see https://docs.oracle.com/javase/8/docs/api/java/security/cert/X509Certificate.html.

Related links

- The OGNL language guide from Apache Commons

Issuance criteria and multiple virtual server IDs

Virtual server IDs offer critical information and functionality in the context of connections.

When you use virtual server IDs to connect to multiple environments in one connection, verifying at runtime the virtual server ID in conjunction with other end-user attributes, such as group membership, protects against unauthorized access.

For instance, both the sales and the support departments of contoso.com, the identity provider (IdP), have their own departmental subdomains, sales.contoso.com and support.contoso.com. The service provider (SP) identifies both environments under the parent domain, contoso.com.

In this scenario, you can configure the PingFederate IdP server to include both sales.contoso.com and support.contoso.com as the virtual server IDs in the SP connection.

If you use one IdP adapter to authenticate end users from both departments, use an OGNL expression to cross-check the virtual server ID information in the request and the end user's group membership information.

```
#this.get("ds.memberOf")!=null?
(
  (
    #this.get("ds.memberOf").toString().matches("(?i)CN=Eng,OU=E,DC=contoso,DC=com")
    &&
    #this.get("context.VirtualServerId").toString()=="Engineering"
  )||
  (
    #this.get("ds.memberOf").toString().matches("(?i)CN=Mkt,OU=M,DC=contoso,DC=com")
    &&
    #this.get("context.VirtualServerId").toString()=="Marketing"
  )
):false
```

> **Note**
>
> Line breaks are inserted for readability only. You must enter statements calling methods whose arguments are enclosed in quotes on a single line.

Related links

- Multiple virtual server IDs
- The OGNL language guide from Apache Commons

Expressions for OAuth and OpenID Connect uses cases

You can use OGNL expressions to retrieve various request-attributes through the HTTP Request Java object.

Client authentication method

The following sample expression retrieves the authentication method that a client uses. This sample expression is applicable to all clients.

```
#this.get("context.HttpRequest").getObjectValue().getAttribute("com.pingidentity.oauth.client.authnType")
```

Private key JSON web token (JWT)

In the following sample expressions, the former retrieves a claim value from the private key JWT with which a client authenticates and the latter retrieves the private key JWT itself. They are only applicable to clients using the private_key_jwt authentication method.

Retrieving the aud *claim value*

```
#claims =
#this.get("context.HttpRequest").getObjectValue().getAttribute("com.pingidentity.oauth.client.jwtClaimsMap"),
#claims.get("aud")
```

Retrieving the entire private key JWT

```
#this.get("context.HttpRequest").getObjectValue().getParameter("client_assertion")
```

Related links

- Configuring access token fulfillment
- The OGNL language guide from Apache Commons

Using the OGNL edit window

Access the in-line OGNL editor and test expressions.

About this task

An in-line editor is available for OGNL expressions. The editor validates the expression and allows an administrator to enter input values and test the resulting output.

Steps

- To reach the OGNL editor, click **Edit** under **Actions** for an expression on any of the **Attribute Fulfillment** windows or click **Test** in the **Show Advanced Criteria** section on the **Issuance Criteria** window.

  > **Note**
  >
  > The test function does not work for the `context.httpRequest` attribute because its value is an object rather than text.

- To test an expression:

  1. Enter an input value in the **Value** text box associated with the attribute.
  2. Click the **Test**link near the bottom-right of the window.

     If the expression contains no errors, the result appears under **Test Results**.

     > **Important**
     >
     > If you want to save changes to an expression, click **Update** under **Actions**. To discard changes, click the **Cancel** link under **Actions**. Click the **Cancel** button near the bottom of the window to discard all changes you made in the current task.

Related links

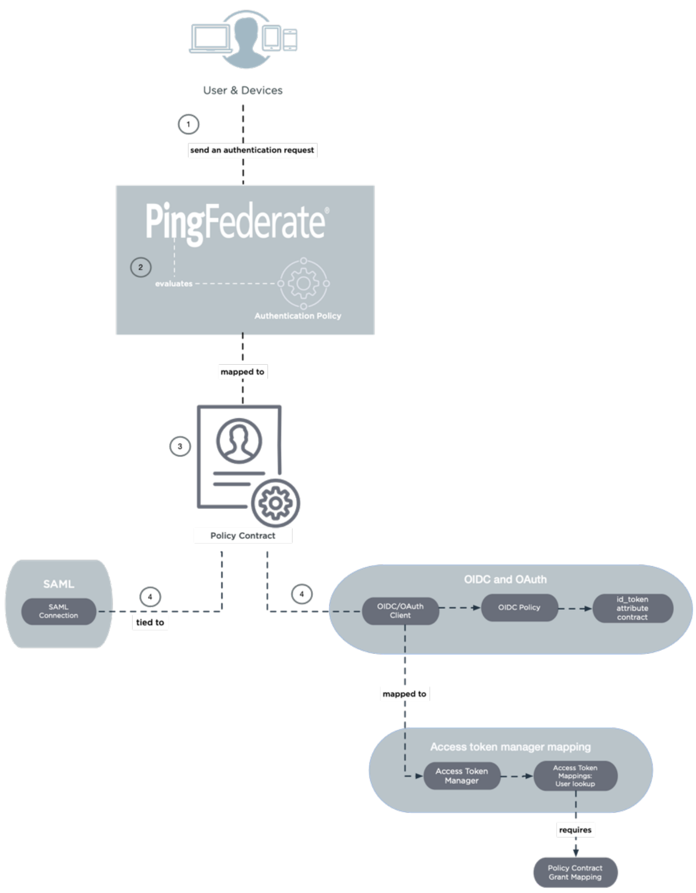
- The OGNL language guide from Apache Commons

Authentication policies

Authentication policies are an optional configuration in PingFederate and help administrators implement complex authentication requirements.

Administrators can configure one or more authentication selector instances to evaluate conditions of the requests and define policies to route the request to a series of approved authentication sources or deny the request based on the results from the authentication selector instances, authentication sources, or both. Administrators can also reuse an authentication policy by ending it with an authentication policy contract or a local identity profile and applying the authentication policy contract in multiple use cases.

OAuth/OIDC and SAML browser authentication flows



Processing steps

1. A client initiated authentication request is sent to PingFederate.
2. PingFederate evaluates the authentication policy, which defines the decision to route a request through a series of approved authentication sources.
3. The authentication policy is mapped to the policy contract.

   The authentication policy determines how the user signs on and drives the authentication experience, such as form-based authentication, Kerberos authentication, or multi-factor authentication (MFA).

> **Note**
>
> PingFederate can enforce authentication policies based on the requesting OAuth client as well as only enforce policy rules for authentication policy contract branches that are mapped to an access token manager (ATM). Learn more in Policies.

4. For an OIDC/OAuth flow, the policy contract checks the attribute contract connected to authentication sources or datastores, or for a SAML connection, the policy contract checks the SAML connection tied to the policy contract.

> **Important**
>
> For an OIDC authentication flow, you must set up an OIDC application in PingFederate. Learn more in Setting up an OIDC application in PingFederate.

5. The authentication request either succeeds or fails based on the results of the policy evaluation and authentication requirements.

Selectors

Authentication selectors provide a plugin capability for PingFederate to evaluate various conditions related to the requests. PingFederate comes bundled with a set of authentication selectors.

As an example, you can create an HTTP Header Authentication Selector to detect mobile browsers, a CIDR Authentication Selector to evaluate whether the users' IP addresses fall within your internal network ranges, or an HTTP Request Parameter Authentication Selector to identify identity provider (IdP) connections based on the `PartnerIdpId` parameter values provided in the service provider (SP)-initiated SSO requests.

Alternatively, you can create custom authentication selectors that suit your needs by using the PingFederate SDK.

> **Tip**
>
> The Javadoc for PingFederate is located in the `<pf_install>/pingfederate/sdk/doc` directory.

Managing authentication selector instances

You can manage authentication selectors on the **Selectors** window in the PingFederate administrative console.

Steps

- Go to **Authentication** > **Policies** > **Selectors**.
- To configure a new instance, on the **Selectors** window, click **Create New Instance**.
- To modify an existing instance, select it by its name under **Instance Name**.
- To remove an existing instance or to cancel the removal request, click **Delete** or **Undelete**.

> **Note**
>
> You can only remove a selector instance if it is not deployed in any authentication policy.

Choosing a selector type

Choose an authentication selector instance from the list of available selector types.

Steps

1. Go to **Authentication** > **Policies** > **Selectors**.
2. Click **Create New Instance**.
3. In the **Instance Name** field, enter an instance name.
4. In the **Instance ID** field, enter an instance ID.
5. From the **Type**list, select the desired type of authentication selector.

Configuring an authentication selector instance

The configuration of an authentication selector instance varies depending on the authentication selectors deployed on your server.

Steps

1. Refer to subsequent topics for configuration steps of each of the bundled authentication selectors.

   Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Configuring the CIDR Authentication Selector

The CIDR Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the IP address of an incoming single sign-on request.

About this task

Use this selector in authentication policies to choose from authentication sources that share a similar level of assurance, such as among multiple HTML Form Adapter instances or between a Kerberos Adapter instance and an X.509 identity provider (IdP) Adapter instance. For example, use this selector in authentication policies to route internal requests to a Kerberos Adapter instance.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, click **Add a new row to 'Networks'** and enter a network range. Click **Update**.

> **Note**
>
> To see the **Add a new row to 'Networks'**option, ensure you have set the **Authentication Selector Instance** type to **CIDR Authentication Selector**on the **Type** tab.

Example:

Sample IPv4 network range

Enter `192.168.101.0/24` to cover 256 IPv4 addresses, ranging from `192.168.101.0` through `192.168.101.255`.

Sample IPv6 network range

Enter `2001:db8::/123` to cover 32 IPv6 addresses, ranging from `2001:db8::` through `2001:db8::1f`.

5. **Optional:** Repeat the previous step to add more network ranges.

> **Note**
>
> Display order does not matter.

> **Tip**
>
> If you want to include all IPv4 addresses for testing, add two separate ranges: `0.0.0.0/1` and `128.0.0.0/1`. The CIDR Authentication Selector interprets a specification of `0.0.0.0/0` as an empty range rather than as a wildcard for all addresses.

Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

6. **Optional:** Enter a **Result Attribute Name** value.

> **Note**
>
> This field provides a means to indicate in the SAML assertion whether a network range was matched during processing; the value is either `Yes` or `No`. Any authentication sources configured as a result of this authentication selector must have their attribute contract extended with the value of the **Result Attribute Name** field to use its value to fulfill an attribute contract or for issuance criteria.

Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Result

When you place this selector instance as a checkpoint in an authentication policy, it forms two policy paths: **Yes** and **No**. If the IP address of an incoming single sign-on (SSO) request matches one of the defined network ranges, the selector returns true. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **Yes**. If the IP address of an incoming SSO request matches none of the defined network ranges, the selector returns false. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **No**.

Configuring the Cluster Node Authentication Selector

The Cluster Node Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the PingFederate cluster node that is servicing the request in authentication policies.

About this task

For example, this selector allows you to choose whether Integrated Windows Authentication (IWA) is attempted based on the PingFederate cluster node with which a Key Distribution Center (KDC) is associated.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** window, select the **Field Value** on which to branch policy paths. The authentication selector provides a means of choosing authentication sources at runtime based on the cluster node on which it is executing.

   ### *Node Index*

   Select **Node Index** to use the `pf.cluster.node.index` value specified in `run.properties`.

   ### *Node Tag*

   Select **Node Tag** to use the `node.tags` values specified in `run.properties`.
5. On the **Selector Result Values** window, specify the relevant node index or node tag values.

   > **Note**
   >
   > Each selector result value forms a policy path when you place this selector instance as a checkpoint in an authentication policy.

   1. In the **Result Values** field, enter a node index or node tag value based on your cluster configuration and click **Add**. This value should correspond to a node index or node tag of one of the engine nodes in the cluster.
   2. **Optional:** Add more values to differentiate criteria for authentication selection.

      > **Note**
      >
      > Display order does not matter.
      > Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.

   Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Configuring the Connection Set Authentication Selector

The Connection Set Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the target service provider (SP) connection used in a single sign-on (SSO) request and SP connections configured within PingFederate.

About this task

This selector allows you to override connection authentication selection on an individual connection basis in one or more authentication policies.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. From the **Type** list, make sure you select **Connection Set Authentication Selector**.
5. Click **Next**. In the **Authentication Selector** window, click **Add a new row to 'Connections'**.
6. From the **Connection** list, select an SP connection and click **Update**.
7. **Optional:** Repeat the previous step to add more connections. Display order does not matter.

   Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.
8. Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Result

When you place this selector instance as a checkpoint in an authentication policy, it forms two **Yes** and **No** policy paths. If the invoking SP connection matches one of the connections from the set, the selector returns true. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **Yes**. If the invoking SP connection matches none of the connections from the set, the selector returns false. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **No**.

Configuring the Extended Property Authentication Selector

The Extended Property Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between a selector result value and an extended property value from the invoking browser-based SSO connections or OAuth client.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.

2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, select a property from the **Extended Property** list.

> **Note**
>
> The extended property is the property that this selector instance should look for from the invoking connection or client, and compare the populated property value, or values if it is a multivalued extended property, against the selector result values defined in this selector instance.

5. On the **Selector Result Values** tab, specify one or more expected result values.
   1. Enter the exact, case-sensitive, value under **Result Values** and click **Add**.
   2. **Optional:** Add more values to differentiate criteria for authentication selection.

      Display order might matter.

      Expected result values are always sorted alphabetically in ascending order here.

      When you place this selector instance as a checkpoint in an authentication policy, each selector result value forms a policy path. The display order of the resulting policy paths matches the display order here, which may impact the policy outcome. When the policy engine reaches this selector instance, the selector starts from top to bottom. As soon as it finds a match, it exits and returns true. The matching mechanism varies, depending on the type of the extended property selected in step 4.

Matching mechanism for single-value extended properties

The selector compares the property value populated in the invoking connection or client against the configured selector result value. When multiple selector result values exist, the selector starts from the top. If the current selector result value is a case-sensitive exact match, it returns true and exits. Otherwise, it moves on to the next selector result value and tries again.For example, assume this selector instance, named ExtProps, is configured with expected result values of `Alpha`, `Bravo`, and `Charlie`. The invoking connection is populated with an extended property value of `Bravo`, and this selector instance is placed as a checkpoint in an authentication policy as follows.

```
ExtProps
+--Alpha
|  <policy path>
|
+--Bravo
|  <policy path>
|
+--Charlie
   <policy path>
```

Given this setup, the selector returns true and exits when it reaches the second selector result value. The policy engine regains control of the request and proceeds with the policy path configured for the selector result value of `Bravo`.

Matching mechanism for multivalued extended properties

The selector compares the property values populated in the invoking connection or client against the configured selector result value. If any one of the property values from the invoking connection or client is a case-sensitive exact match, the selector returns true and exits. When multiple selector result values exist, the selector starts from the top. If the current selector result value is a case-sensitive exact match to any one of the property values from the invoking connection or client, it returns true and exits. Otherwise, it moves on to the next selector result value and tries again.For example, assume the previous selector instance remains. The invoking connection is populated with extended property values of `Alpha` and `Charlie`, and this selector instance remains as a checkpoint in an authentication policy.In this scenario, the selector returns true and exits when it reaches the first selector result value. The policy engine regains control of the request and proceeds with the policy path configured for the selector result value of `Alpha`. Even though `Charlie`, the expected selector result value, is also a case-sensitive exact match to `Charlie`, one of the property values from the invoking connection, because the selector has already exited and returned control to the policy engine when it reaches `Alpha`, the policy engine will never execute the policy path configured for the selector result value of `Charlie`.

Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.

6. Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Example

1. Go to **System > Server > Extended Properties**.
2. On the **Extended Properties**window, define a multivalued extended property, and name it `configStatus`.
3. Create an SP connection with the following characteristics:
   - On the **Extended Properties** window, add two values for the `configStatus` extended property: `DEV` and `TEST`.
   - On the **Attribute Source Mapping** window, map an authentication policy contract to the service provider (SP) connection. The policy contract name is `APC`.
4. Create an instance of the Extended Property Authentication Selector with the following characteristics:
   - On the **Type** tab, name the selector instance `ExProps`.
   - On the **Authentication Selector** tab, select `configStatus` from the list.
   - On the **Selector Result Values** tab, enter `DEV` and `TEST`.
5. Create and activate the following identity provider (IdP) authentication policy.

```
ExtProps
+--DEV
|  OpenToken
|  +--Fail: Done
|  +--Success: APC
|
+--TEST
   HTML
   +--Fail: Done
   +--Success: APC
```

Configure each `APC` to fulfill values obtained from its preceding adapter instance.

When processing SSO requests intended for this SP connection, because the policy engine is able to match one of the populated property values, `DEV`, from the SP connection to the first selector result value, also `DEV`, it will always invoke the OpenToken IdP Adapter instance based on the `DEV` policy path. The `TEST` policy path is never executed for this SP connection.

On the other hand, if you remove `DEV`, an extended property value, from the SP connection, the policy engine will route SSO requests intended for this SP connection to the HTML Form Adapter instance based on the `TEST` policy path. The `DEV` policy path is never executed for this SP connection.

Related links

- Extended properties

Configuring the HTTP Header Authentication Selector

The HTTP Header Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found in a specified HTTP header.

About this task

Use this selector in one or more authentication policies to choose from authentication sources that share a similar level of assurance, such as among multiple HTML Form Adapters or between a Kerberos Adapter and an X.509 Adapter. For example, use this selector to choose an authentication source based on the user's browser identified by the `User-Agent` HTTP header.

> **Important**
>
> Don't use this selector to determine whether an authentication source with a higher level of assurance should be bypassed because HTTP request headers could potentially be forged.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.

2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.

3. On the **Type** tab, configure the basics of this authentication selector instance.

4. On the **Authentication Selector** tab, click **Add a new row to 'Results'**.

5. Enter an expression for use when inspecting the HTTP header value of the target HTTP header under **Match Expression**, and click **Update**.

   > **Note**
   >
   > Wildcard entries are allowed, such as `*value*`.

6. (Optional) Repeat the previous step to add more expressions. Display order does not matter.

   > **Note**
   >
   > Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

7. In the **Header Name** field, enter the type of HTTP header you want the selector to inspect. This field isn't case-sensitive.

8. (Optional) To disable case-sensitive matching between the HTTP header values from the requests and the **Match Expression** values specified on this window, clear the **Case-Sensitive Matching** checkbox.

   The **Case-Sensitive Matching** checkbox is selected by default.

9. Complete the configuration. On the **Summary** tab, click **Done**. On the **Selectors** window, click **Save**.

Result

When you place this selector instance as a checkpoint in an authentication policy, it forms two policy paths: **Yes** and **No**. If the value of the specified HTTP header matches one of the configured values, the selector returns true. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **Yes**. If the value of the specified HTTP header matches none of the configured values, the selector returns false. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **No**.

Example

To detect the most common browsers based on the `User-Agent` HTTP request header, configure an HTTP Header Authentication Selector instance as follows.

1. Enter these entries under **Match Expression**.

Browser	Expression
Chrome	`*Chrome*`
Firefox	`*Firefox*`
Safari	`*Safari*`

2. In the **Header Name** field, enter `User-Agent`.

Configuring the HTTP Request Parameter Authentication Selector

The HTTP Request Parameter Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on query parameter values.

About this task

Use this selector in one or more authentication policies to choose from authentication sources that share a similar level of assurance, such as among multiple instances of the HTML Form Adapter or between a Kerberos Adapter instance and an X.509 Adapter instance. For example, use an instance of this selector to choose an authentication experience based on the reward program information indicated by a query parameter in the single sign-on (SSO) request.

> **Important**
>
> Do not use this selector to determine whether an authentication source with a higher level of assurance should be bypassed because query parameters could potentially be forged.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, configure the applicable selector instance settings.
   1. Enter the exact, case-sensitive name of the request parameter in the **HTTP Request Parameter Name** field.

      > **Important**
      >
      > The policy engine is capable of tracking HTTP request parameters that it receives from the initial request and making them available to selector instances throughout the policy. If you plan on using this selector instance as the second, or subsequent, checkpoint in at least one authentication policy, add the **HTTP Request Parameter Name** value on the **Tracked HTTP Parameters** window. For more information, see Defining authentication policies.

   2. **Optional:** To disable case-sensitive matching between the HTTP request parameter values from the requests and the **Match Expression** values specified on the **Selector Result Values** window, clear the **Case-Sensitive Matching** check box.

      > **Note**
      >
      > The **Case-Sensitive Matching** check box is selected by default.

   3. **Optional:** Enable policy paths to handle additional scenarios.

      For more information, see the following table.

Field	Description
Enable 'Any' Result Value	Each configured selector result value forms a separate authentication policy path. Select this check box if you want to enable a single policy path for the scenario where the HTTP request parameter value matches any one of the configured selector result values. This check box is not selected by default.
Enable 'No Match' Result Value	Selector evaluation fails and the next applicable authentication policy is executed when the HTTP request parameter value does not match any of the configured selector result values. Select this check box if you want to enable a policy path to handle this scenario. This check box is not selected by default.
Enable 'Not in Request' Result Value	Selector evaluation fails and the next applicable authentication policy is executed if the HTTP request parameter is not found. Select this check box if you want to enable a policy path to handle this scenario. This check box is not selected by default.

5. On the **Selector Result Values** window, enter a request parameter value under **Result value**, and then click **Add**.

> **Note**
>
> Wildcard entries are allowed, such as `*value*`.

> **Important**
>
> A more specific match is a better match, and an exact match is the best match.

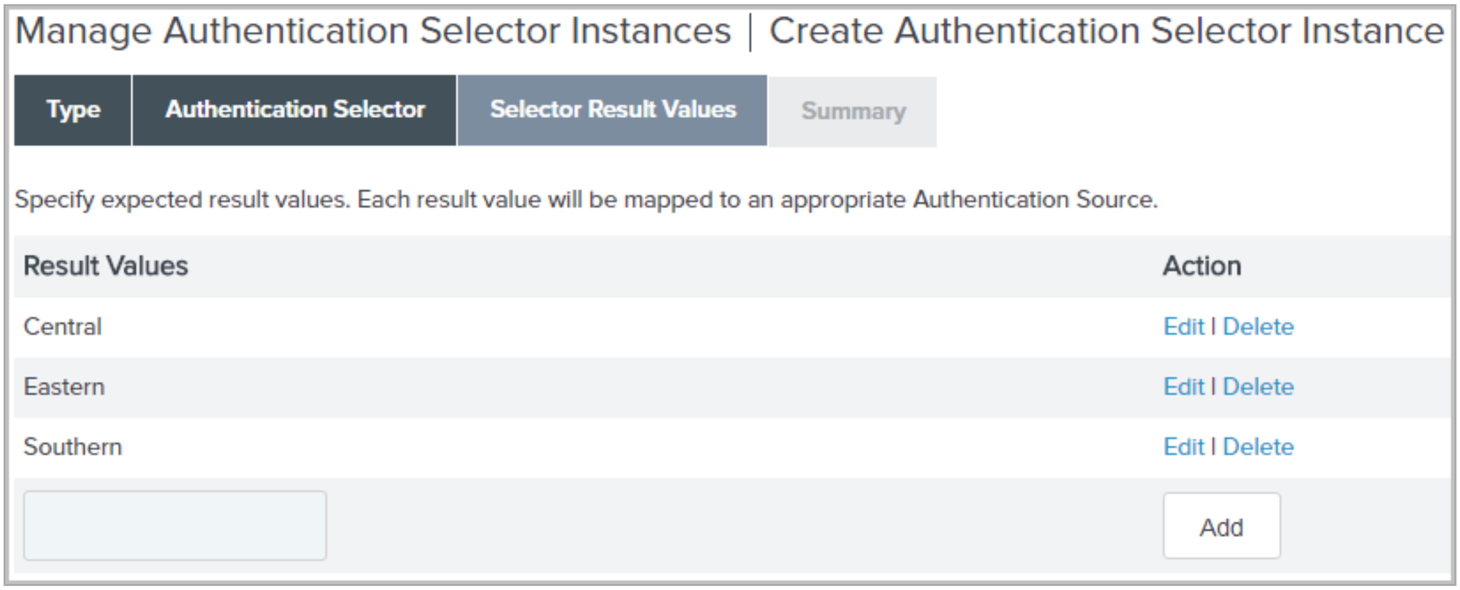
6. **Optional:** Repeat the previous step to add more request parameter values. Display order does not matter.

> **Note**
>
> If you have not enabled the **Any** policy path in step 4c, each selector result value forms a policy path when you place this selector instance as a checkpoint in an authentication policy.
> If you have enabled the **Any** policy path, only one policy path is formed.
> Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.

7. Complete the configuration.
   1. On the **Summary** tab, click **Done**.
   2. On the **Selectors** window, click **Saved**.

Example

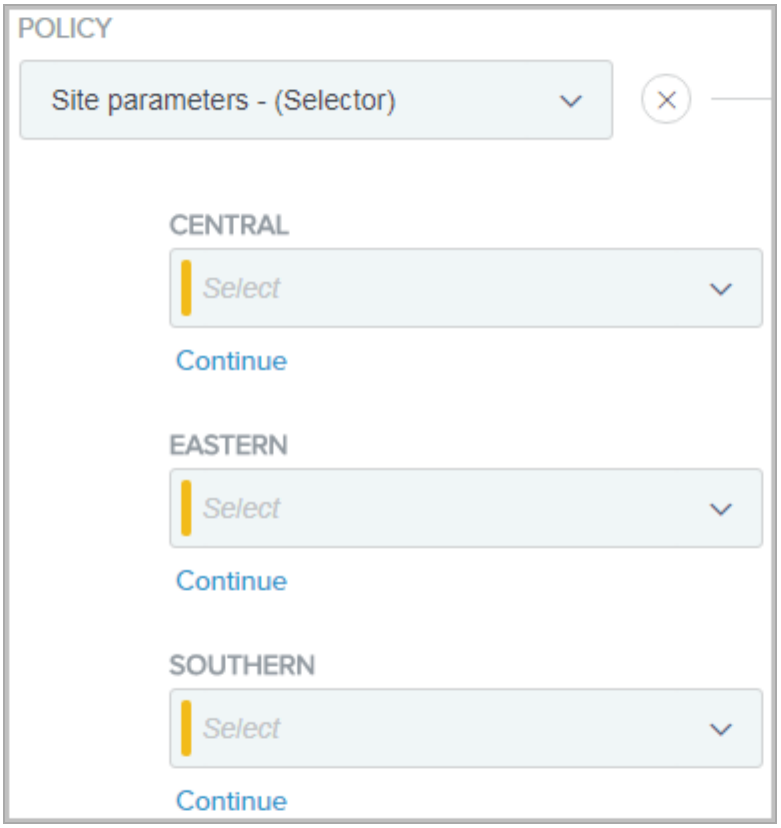
Suppose you enter three selector result values, `Central`, `Eastern`, and `Southern`, on the **Selector Result Values** window, as illustrated in the following screen capture.


Manage Authentication Selector Instances | Create Authentication Selector Instance

Specify expected result values. Each result value will be mapped to an appropriate Authentication Source.

Result Values	Action
Central	Edit I Delete
Eastern	Edit I Delete
Southern	Edit I Delete
	Add

If you have not enabled any additional policy paths in step 4c, as you place this selector instance as a checkpoint in an authentication policy, three policy paths are extended from the selector instance, one for each of the configured selector result values.



Configuring the OAuth Client Set Authentication Selector

The OAuth Client Set Authentication Selector selector allows you to override client authentication select on an individual client basis in one or more authentication policies.

About this task

The selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the client information in an OAuth request and the OAuth clients configured in the PingFederate OAuth authorization server (AS).

> **Note**
>
> The OAuth Client Set Authentication Selector is only applicable to OAuth clients using the authorization code, device authorization grant, or implicit flow.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, click **Add a new row to 'Clients'**.

   > **Note**
   >
   > If you don’t see **Add a new row to 'Clients'**, go back to the **Type** tab and ensure you have selected **OAuth Client Set Authentication Selector** from the **Type** list.

5. From the **Client ID** list, select an OAuth client and click **Update**.
6. **Optional:** Repeat the previous step to add more clients.

   Display order does not matter.

7. Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

   On the **Summary** tab, click **Done**. On the **Selectors** window, click **Save**.

Result

When you place this selector instance as a checkpoint in an authentication policy, it forms two policy paths: **Yes** and **No**. If the invoking client matches one of the clients from the set, the selector returns true. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **Yes**. If the invoking client matches none of the clients from the set, the selector returns false. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **No**.

Configuring the OAuth Scope Authentication Selector

The OAuth Scope Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the scopes of an OAuth authorization request and scopes configured in the PingFederate OAuth authorization server (AS).

Before you begin

Go to **System** > **OAuth Settings** > **Authorization Server Settings** and configure one or more scopes.

About this task

This selector allows you to control the strength of authentication based on client access requirements. For example, if a client requires write access to a resource, you can deploy an instance of the OAuth Scope Authentication Selector in one or more authentication policies to choose an adapter that offers a stronger form of authentication, such as the X.509 client certificate, instead of username and password.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, select the required scopes, scope groups, or both.

> **Note**
>
> Both common and exclusive scopes are available for selection.

> **Important**
>
> This selector matches only scopes from OAuth authorization requests to the authorization endpoint, `/as/authorization.oauth2`. SAML single sign-on (SSO) requests do not match this authentication selector's criteria and result in a returned result value of `No`. If you are using this selector and selectors specific to SAML connections, list this selector first in the mapping list so that it takes precedence for OAuth without disrupting selector logic on SAML connections.

5. Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Result

When you mark this selector instance as a checkpoint in an authentication policy, it forms two policy paths: **Yes** and **No**. If the requested scopes satisfy all the selected scopes, the selector returns true. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **Yes**. If the requested scopes do not satisfy all the selected scopes, the selector returns false. The policy engine regains control of the request and proceeds with the policy path configured for the result value of **No**.

Related links

- Scopes and scope management

Configuring the Requested AuthN Context Authentication Selector

The Requested AuthN Context Authentication Selector enables PingFederate to choose configured authentication sources or other selectors.

About this task

This selector chooses authentication sources or selectors based on the authentication contexts requested by a service provider (SP) for browser single sign-on (SSO) requests, or a relying party (RP) for OAuth with OpenID Connect (OIDC) use cases in authentication policies.

For browser SSO, this authentication selector works in conjunction with SP connections with Security Assertion Markup Language (SAML) 2.0 only, using the SP-initiated SSO profile. Other browser SSO protocols do not support authentication context. For OAuth, clients supporting the OIDC protocol must include the optional `acr_values` parameter in their authorization requests to indicate their preferred authentication context, or contexts.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** tab, configure the applicable selector instance settings:
    1. Select the **Add or Update AuthN Context Attribute** check box if you want to update the authentication context attribute value with the value specified in the **Selector Result Values** tab.

       Result:

       When selected, which is the default, the check box on this window provides a means to:

       - Add the value of the authentication context determined by the selector into the SAML assertion.
       - When applicable, replace any value returned from the associated adapter instance with the selector result value.
    2. **Optional:** Select the **Override AuthN Context for Flow** check box to allow the authentication selector result value to override the authentication context value for the entire policy flow.

       > **Note**
       >
       > This check box is only available when the **Add or Update AuthN Context Attribute** check box is selected.

       Result:

       When selected, which is the default for fresh installations, the selector result will determine the authentication context value for the entire flow and override any subsequently invoked authentication sources and their authentication context values. This authentication selector result value takes precedence and determines the authorization context in the outgoing assertion or ID token.
    3. **Optional:** Enable policy paths to handle additional scenarios.

For more information, refer to the following table.

Field	Description
Enable 'No Match' Result Value	Selector evaluation fails and the next applicable authentication policy is executed if the requested authentication context does not match any of the configured selector result values. Select this check box if you want to enable a policy path to handle this scenario. This check box is not selected by default.
Enable 'Not in Request' Result Value	Selector evaluation fails and the next applicable authentication policy is executed if no requested authentication context is found. Select this check box if you want to enable a policy path to handle this scenario. This check box is not selected by default.

5. In the **Selector Result Values** window, specify the authentication contexts to use as the criteria:

   1. Enter the exact, case-sensitive parameter value under **Result Values**, and then click **Add**.

      > **Note**
      >
      > The value can include URIs defined in Authentication Context for the OASIS Security Assertion Markup Language (SAML) 2.0 or any other value agreed upon with the partner.

   2. **Optional:** Add more values to differentiate criteria for authentication selection.

      Display order does not matter.

      Each selector result value forms a policy path when you place this selector instance as a checkpoint in an authentication policy (regardless of whether you have enabled the **No Match** or **Not in Request** policy path in step 4c).

      Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.

6. Complete the configuration.

   1. On the **Summary** tab, click **Done**.

   2. On the **Selectors** window, click **Save**.

Configuring the Session Authentication Selector

The Session Authentication Selector enables PingFederate to choose a policy path at runtime based on whether the user already has a PingFederate authentication session for a particular source.

Steps

1. Go to **Authentication** > **Policies** > **Selectors** to open the **Selectors** window.
2. On the **Selectors** window, click **Create New Instance** to start the **Create Authentication Selector Instance** workflow.
3. On the **Type** tab, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** window, click **Add a new row to 'Authentication Sources'**.
5. Select an IdP adapter instance or an IdP connection from the list, enter a value under **Result Value** for the selected authentication source, then click **Update**.

   The **Result Value** field controls the label shown for the policy path created by the selected authentication source.

   > **Note**
   >
   > You must enable authentication sessions for the selected authentication source, or globally for all authentication sources, on the**Sessions** window. Click **Manage Sessions** to review and configure authentication sessions.

6. **Optional:** Repeat the previous step to add more authentication sources.

   Display order might matter.

   When you place this selector instance as a checkpoint in an authentication policy, each selector result value forms a policy path. The display order of the resulting policy paths matches the display order here, which may impact the policy outcome. When the policy engine reaches this selector instance, the selector starts from top to bottom. It exits and returns true as soon as it finds a match.

   As needed, use the up and down arrows to re-arrange the display order here, which also re-prioritizes the resulting policy paths.

   In addition, when no session exists for any of the defined sources, the result value for the first authentication source is returned unless the **Enable 'No Session' Result Value** check box is selected, in which case an additional policy path is added as the last path when this selector instance is placed as a checkpoint in an authentication policy.

   Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

7. **Optional:** Select the **Enable 'No Session' Result Value** check box to create a separate policy path for the scenario where no session exists for any of the defined sources.

   This check box is not selected by default.

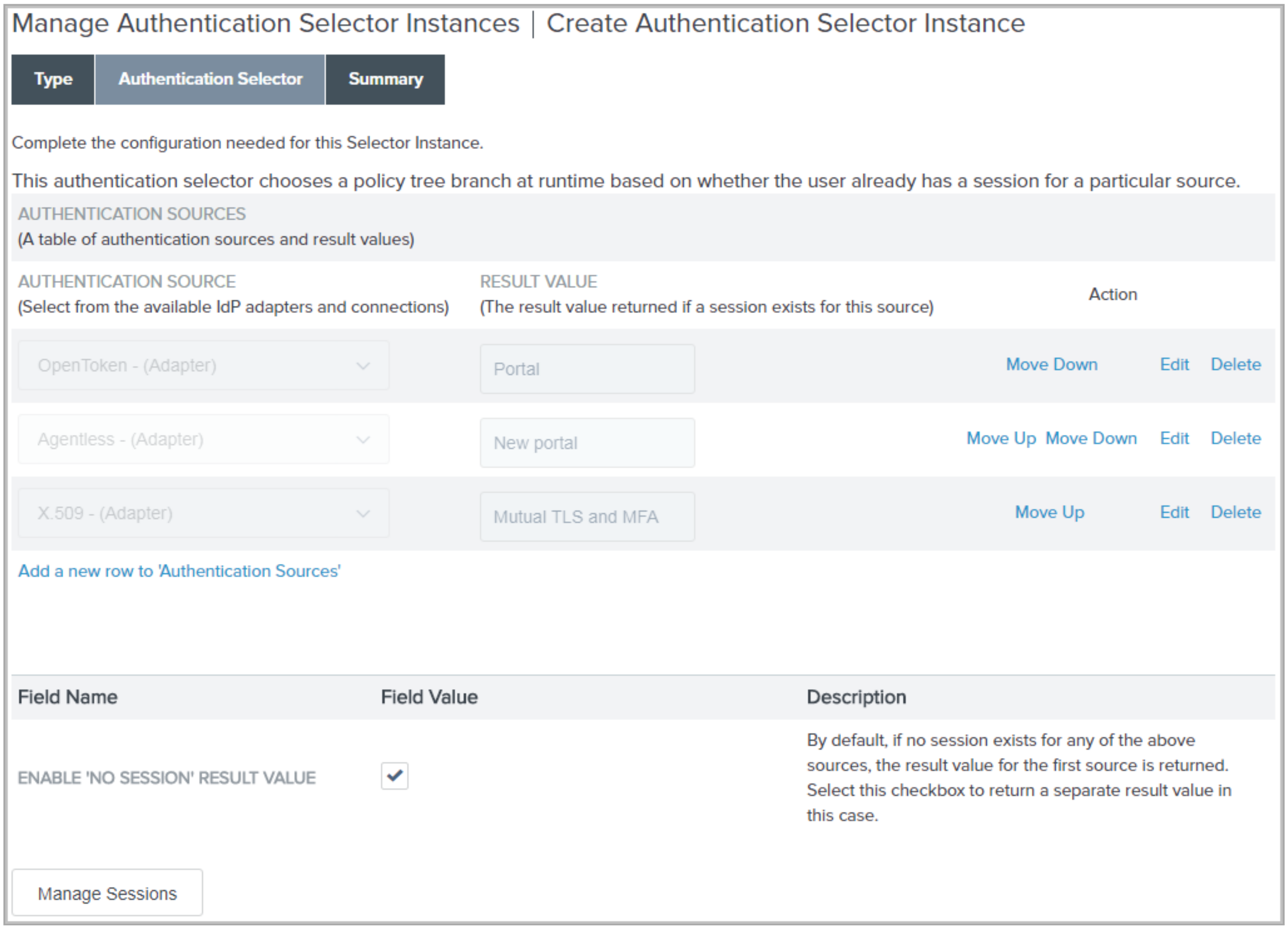
8. Complete the configuration.On the **Summary** tab, click **Done**.On the **Selectors** window, click **Save**.

Result

When you place this selector instance as a checkpoint in an authentication policy, each selector result value forms a policy path that you can define the desired authentication experience and requirements.

Example

The following screen capture illustrates a configuration where three authentication sources are defined and the **Enable 'No Session' Result Value** check box is selected.



Manage Authentication Selector Instances | Create Authentication Selector Instance

Complete the configuration needed for this Selector Instance.

This authentication selector chooses a policy tree branch at runtime based on whether the user already has a session for a particular source.

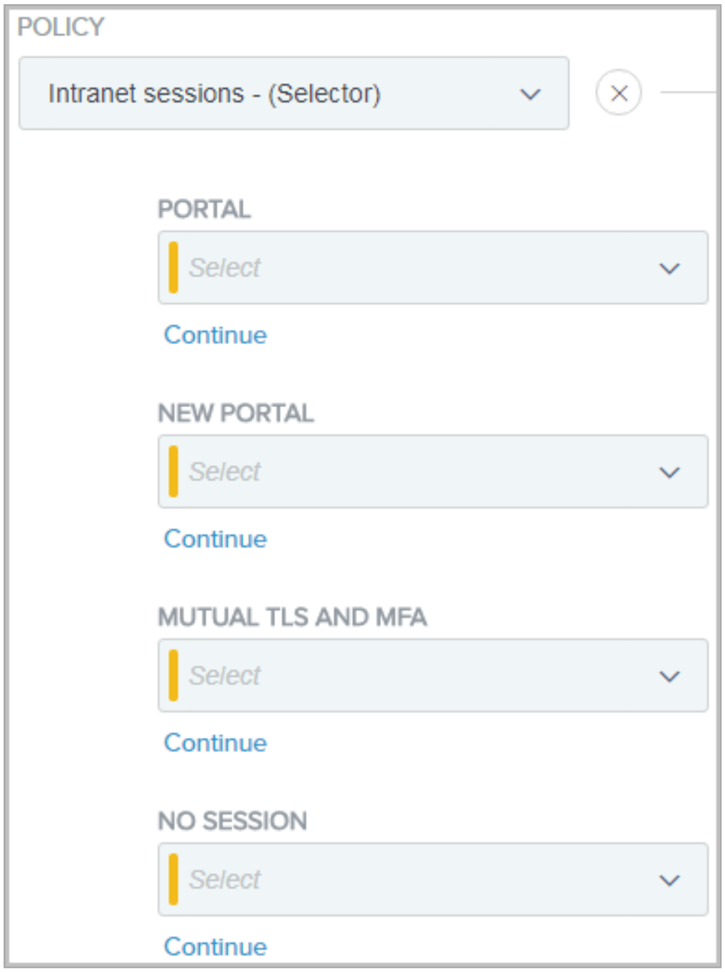
AUTHENTICATION SOURCES
(A table of authentication sources and result values)

AUTHENTICATION SOURCE (Select from the available IdP adapters and connections)	RESULT VALUE (The result value returned if a session exists for this source)	Action
OpenToken - (Adapter)	Portal	Move Down Edit Delete
Agentless - (Adapter)	New portal	Move Up Move Down Edit Delete
X.509 - (Adapter)	Mutual TLS and MFA	Move Up Edit Delete

Add a new row to 'Authentication Sources'

Field Name	Field Value	Description
ENABLE 'NO SESSION' RESULT VALUE	☑	By default, if no session exists for any of the above sources, the result value for the first source is returned. Select this checkbox to return a separate result value in this case.

When this selector instance (named Intranet sessions) is placed in a policy, four policy paths are formed.



Configuring a sample use case

Use the following sample setup to configure one of the common use cases where you have two categories of service providers (SPs).

Before you begin

For this sample use case, you must have the following components:

- An authentication policy contract
- Multiple SP connections. All connections use the same authentication policy contract as their sole authentication source
- Instances of the required adapters
- An instance of the Connection Set Authentication Selector to isolate high-value connections from the rest of the connections

About this task

The Session Authentication Selector lets PingFederate choose a policy path at runtime based on whether the user already has a PingFederate authentication session for a particular source.

You need to enforce authentication requirements on two categories of service provider connections:

- For high-value connections, users must authenticate using the X.509 Adapter followed by the PingID Adapter.

- For low-value connections, users can authenticate using the HTML Form Adapter or the X.509 Adapter followed by the PingID Adapter.

To fulfill this use case, follow these configuration steps.

Steps

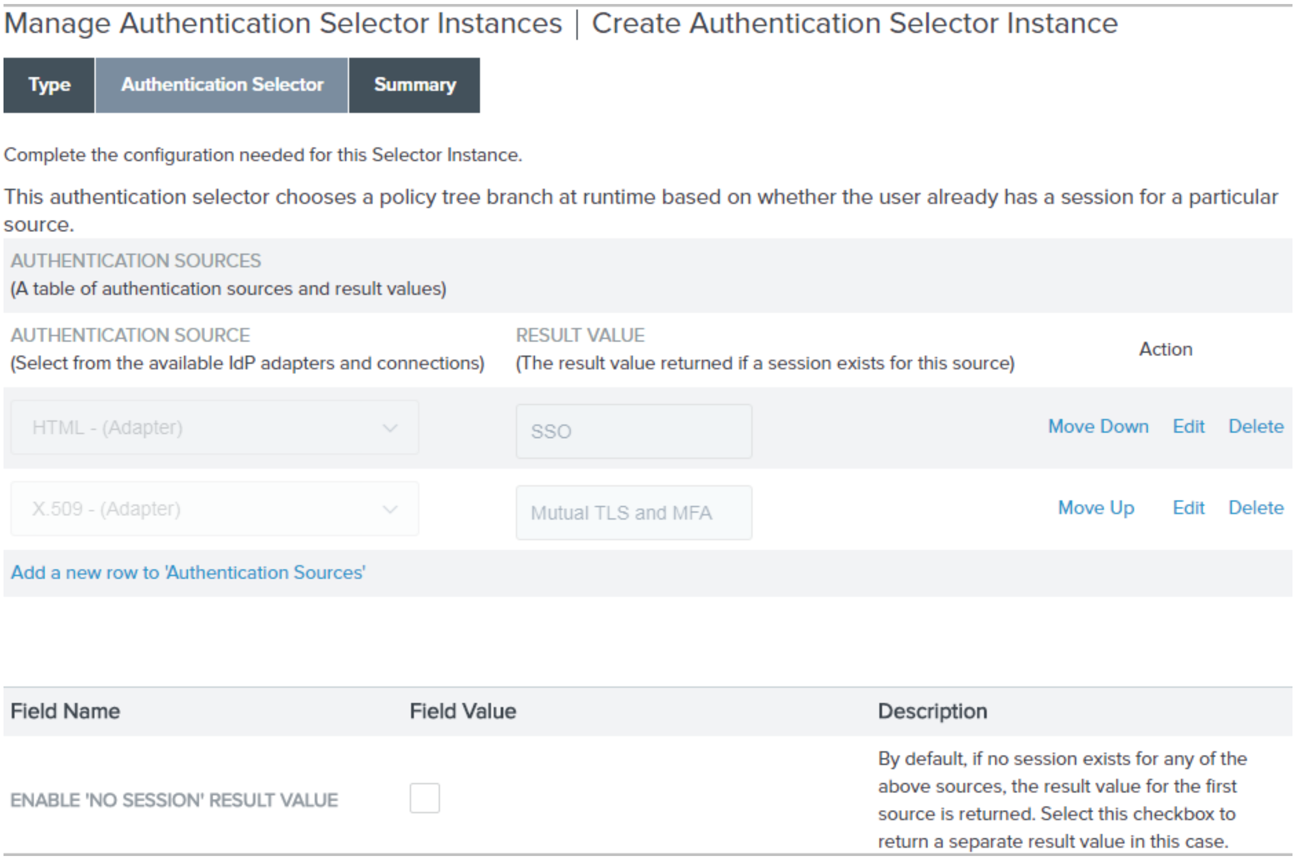
1. Go to **Authentication** > **Policies** > **Selectors**.
2. Create an instance of the Session Authentication Selector to account for authentication sessions acceptable for low-value connections.

   1. Click **Create New Instance**.
   2. On the **Type** tab, enter a name (for example, Sessions for low-value connections) and an ID. Then select **Session Authentication Selector** from the list.
   3. On the **Authentication Selector** tab, leave the **Enable 'No Session' Result Value** check box clear; then configure the following authentication source-to-result value entries.

Authentication source (adapter instance name)	Result value (policy path label)
HTML	**SSO**
X.509	**Mutual TLS and MFA**

Example:

The following screen capture illustrates the setup.



Manage Authentication Selector Instances | Create Authentication Selector Instance

Complete the configuration needed for this Selector Instance.

This authentication selector chooses a policy tree branch at runtime based on whether the user already has a session for a particular source.

AUTHENTICATION SOURCES
(A table of authentication sources and result values)

AUTHENTICATION SOURCE (Select from the available IdP adapters and connections)	RESULT VALUE (The result value returned if a session exists for this source)	Action
HTML - (Adapter)	SSO	Move Down Edit Delete
X.509 - (Adapter)	Mutual TLS and MFA	Move Up Edit Delete

Add a new row to 'Authentication Sources'

Field Name	Field Value	Description
ENABLE 'NO SESSION' RESULT VALUE	□	By default, if no session exists for any of the above sources, the result value for the first source is returned. Select this checkbox to return a separate result value in this case.

   4. On the **Summary** tab, click **Done**.

   5. On the **Manage Authentication Selector Instances** window, click **Save** to keep the newly configured authentication selector instance.

3. Go to **Authentication** > **Policies** > **Policies**.

4. On the **Policies** window, define an authentication policy for high-value connections.

   1. Click **Add Policy**.

   2. In the **Name** field, enter a name for the policy, such as `High-value connections`.

   3. From the **Policy** list, select the instance of the Connect Set Authentication Selector that isolates high-value connections from the rest.

   4. For the **No** policy path, select **Continue**.

   5. For the **Yes** policy path, select the X.509 Adapter instance.

   6. For the **X.509 Adapter instance** > **Fail** policy path, select **Done**.

   7. For the **X.509 Adapter instance** > **Success** policy path, select the PingID Adapter instance.

   8. Below the PingID Adapter instance, click **Options**.

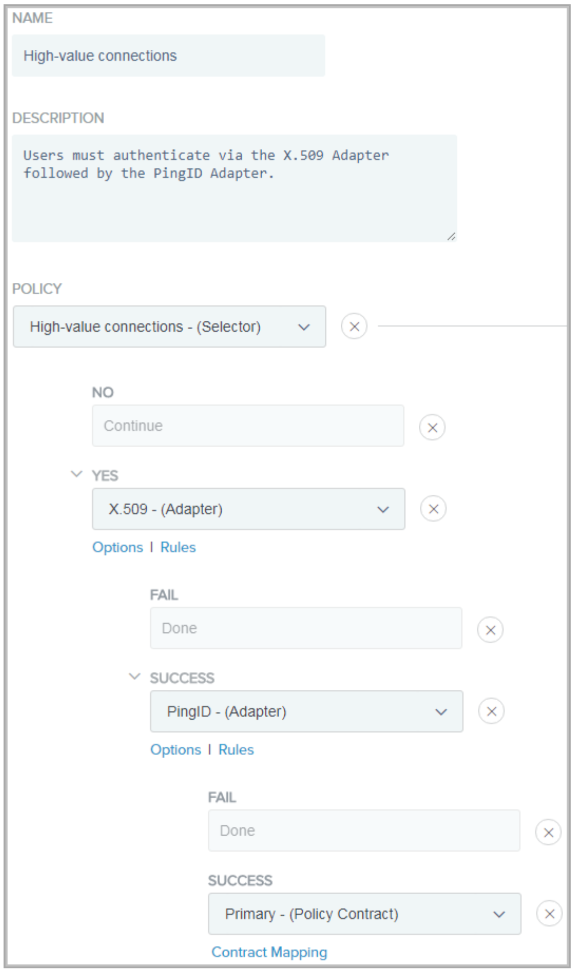
   9. On the **Incoming User ID** window, select the X.509 Adapter instance as the source and `username` as the attribute.

> **Tip**
>
> This step applies only to adapters that support a user identifier to be passed in from an earlier authentication source. The PingID Adapter requires this user identifier. For more information, see Specifying incoming user IDs.

10. For the **X.509 Adapter instance > Success > PingID Adapter instance > Fail** policy path, select **Done**.
11. For the **X.509 Adapter instance > Success > PingID Adapter instance > Success** policy path, select the authentication policy contract.
12. Complete the contract mapping for the authentication policy contract.

    Example:

    The following illustrates the policy created for high-value connections.

    NAME
    High-value connections

    DESCRIPTION
    Users must authenticate via the X.509 Adapter followed by the PingID Adapter.

    POLICY
    High-value connections - (Selector)

    NO
    Continue

    YES
    X.509 - (Adapter)
    Options | Rules

    FAIL
    Done

    SUCCESS
    PingID - (Adapter)
    Options | Rules

    FAIL
    Done

    SUCCESS
    Primary - (Policy Contract)
    Contract Mapping

13. Click **Done**.

5. Define an authentication policy for low-value connections.

    1. Click **Add Policy**.

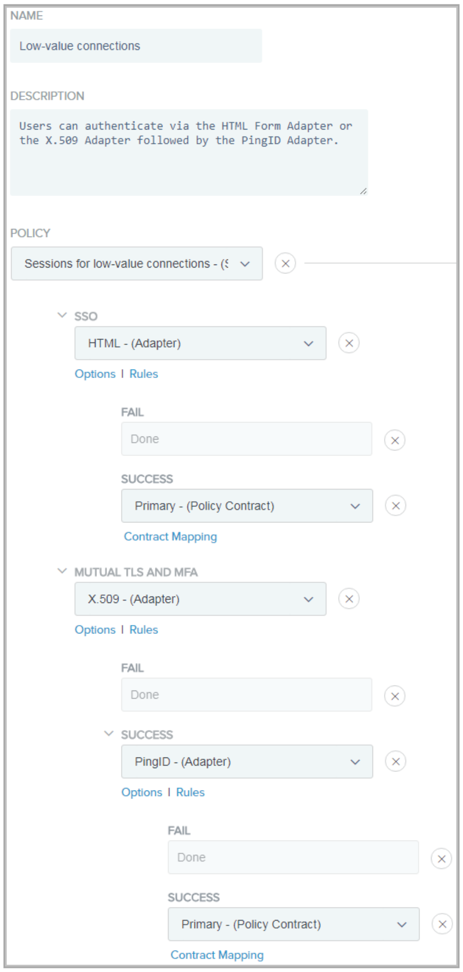
2. Enter a name for the policy, such as Low-value connections.
3. From the **Policy** list, select the instance of the Session Authentication Selector. For more information, see step 2.
4. For the single sign-on (**SSO**) policy path, select the HTML Form Adapter instance.
5. For the **HTML Form Adapter instance > Fail** policy path, select **Done**.
6. For the **HTML Form Adapter instance > Success** policy path, select the authentication policy contract.
7. Complete the contract mapping for the authentication policy contract.
8. For the **Mutual TLS and MFA** policy path, select the X.509 Adapter instance.
9. For the **X.509 Adapter instance > Success** policy path, select the PingID Adapter instance.
10. Below the PingID Adapter instance, click **Options**. Select the X.509 Adapter instance as the source and `username` as the attribute on the**Incoming User ID** window.

    > **Tip**
    >
    > This step only applies to adapters that support a user identifier to be passed in from an earlier authentication source. The PingID Adapter requires this user identifier. For more information, see Specifying incoming user IDs.

11. For the **X.509 Adapter instance > Success > PingID Adapter instance > Fail** policy path, select **Done**.
12. For the **X.509 Adapter instance > Success > PingID Adapter instance > Success** policy path, select the authentication policy contract.
13. Complete the contract mapping for the authentication policy contract.

    Example:

    The following illustrates the policy created for low-value connections.



NAME

Low-value connections

DESCRIPTION

```
Users can authenticate via the HTML Form Adapter or
the X.509 Adapter followed by the PingID Adapter.
```

POLICY

Sessions for low-value connections - (

SSO

HTML - (Adapter)

Options | Rules

FAIL

Done

SUCCESS

Primary - (Policy Contract)

Contract Mapping

MUTUAL TLS AND MFA

X.509 - (Adapter)

Options | Rules

FAIL

Done

SUCCESS

PingID - (Adapter)

Options | Rules

FAIL

Done

SUCCESS

Primary - (Policy Contract)

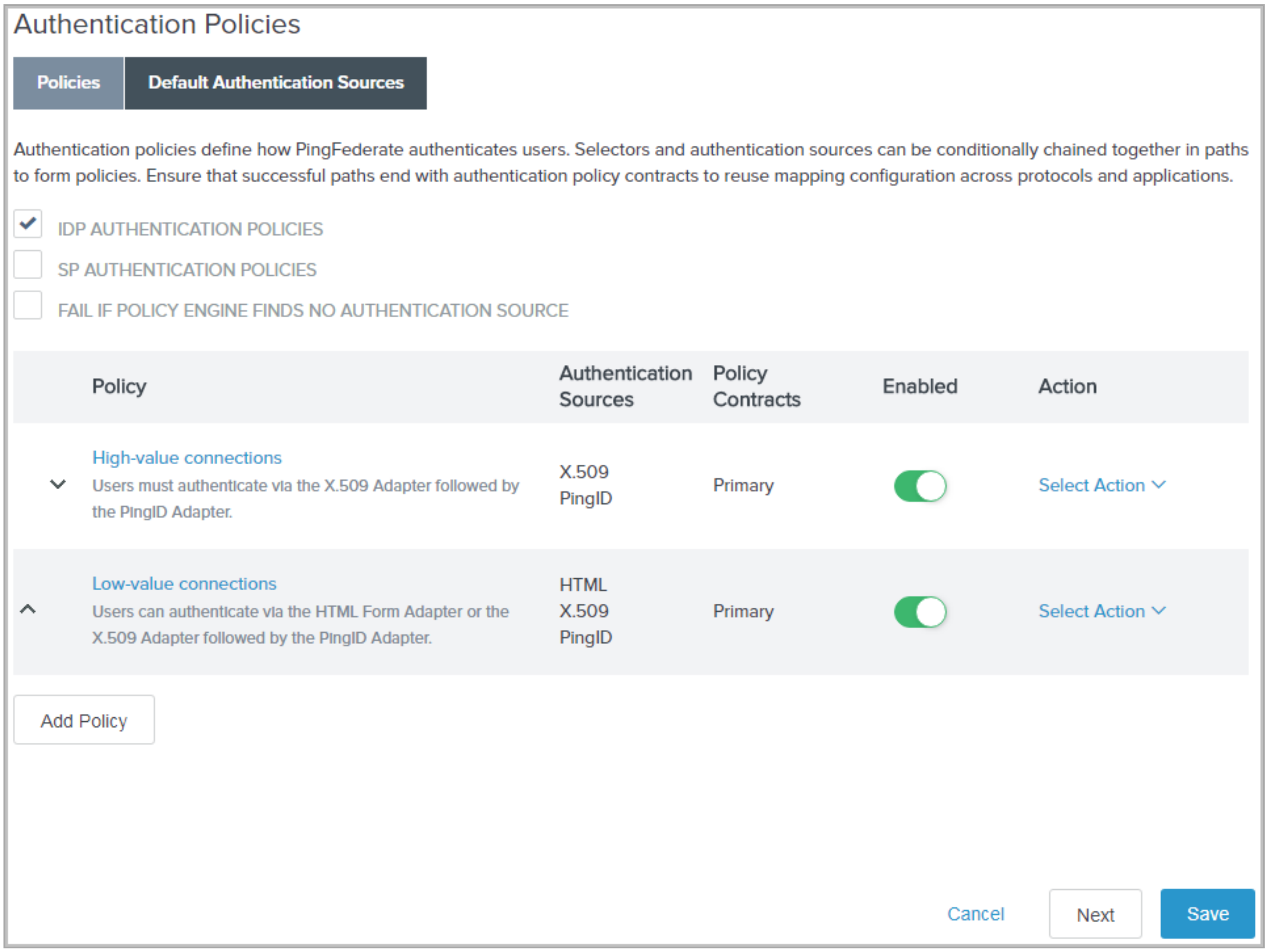
Contract Mapping

14. Click **Done**.

15. To activate authentication polices for identity provider (IdP) browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows, select the **IdP Authentication Policies** check box.

    Example:

    The following screen capture illustrates the policies created this sample use case.



Authentication Policies

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☑ IDP AUTHENTICATION POLICIES

□ SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
High-value connections Users must authenticate via the X.509 Adapter followed by the PingID Adapter.	X.509 PingID	Primary	●	Select Action
Low-value connections Users can authenticate via the HTML Form Adapter or the X.509 Adapter followed by the PingID Adapter.	HTML X.509 PingID	Primary	●	Select Action

6. To keep the newly configured authentication policies, click **Save**.

Policies

You can configure policies, fragments, selectors, policy contracts, and sessions under **Authentication > Policies**.

An authentication policy is a tree of authentication sources, selector instances, or a combination of them, that defines the decision to route a request through a series of approved authentication sources with an optional authentication policy contract or a local identity profile at the end or to deny the request.

Administrators can enable authentication policies on identity provider (IdP) browser single sign-on (SSO) requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows. Administrators can also enable authentication policies on service provider (SP)-initiated Browser SSO requests received at the `/sp/startSSO.ping` endpoint. Individual policies can be disabled, as needed.

The order of authentication policies matters because the policy engine starts from the first policy and works its way down. At runtime, the policy engine derives an authentication tree from the applicable policies and either approves or denies a request.

Policy paths, authentication policy contracts, and local identity profiles

Policy paths

An authentication policy starts with either a selector instance or an authentication source. Authentication sources and most selectors have two results, **Success** or **Fail**, **Yes** or **No**. Each result forms a policy path.

A policy path is open-ended if it contains only one or more selector instances, without any authentication sources. In this scenario, the policy engine continues to the next applicable authentication policy, if any.

+

A policy path is closed-ended if it contains one or more authentication sources, with or without any selector instances. A closed-ended path can optionally end with an authentication policy contract or a local identity profile.

+

> **Note**
>
> A policy path is also closed-ended if it ends with an instance of a custom authentication selector that returns an IdP adapter instance ID or the connection ID of an IdP connection. Because the custom selector returns an authentication source, such a closed-ended path cannot end with an authentication policy contract or a local identity profile. Instead, it must end with an action of **Done** or **Restart**.

Authentication policy contracts and local identity profiles

An authentication policy contract can harness attribute values obtained from all authentication sources along the path leading up to it. Administrators can select the same authentication policy contract or local identity profile for different closed-ended paths, in one or more authentication policies, and fulfill them differently to suit the requirements. To enforce the same set of authentication policies in multiple use cases, map the authentication policy contract to the applicable Browser SSO connections and OAuth grant-mapping configuration.

A policy becomes more complex as the number of paths grows with the number of authentication sources and selector instances.

Multiple policies and runtime behavior

A complex policy can cover a lot of ground. However, depending on the authentication requirements, administrators can also create multiple policies to suit their needs.

When a request arrives at PingFederate, the policy engine skips all disabled policies and any closed-ended paths that are inapplicable to the request. A closed-ended path is considered inapplicable to a request in any of the following conditions:

- The local identity profile at the end of a path is associated with an authentication policy contract that is not mapped to the invoking use case or is blocked by the virtual server ID included in the request.
- The authentication policy contract at the end of a path is not mapped to the invoking use case or is blocked by the virtual server ID included in the request.
- The last authentication source at the end of a path, that does not end with an authentication policy contract or a local identity profile, is not mapped to the invoking use case or is blocked by the virtual server ID included in the request.

> **Note**
>
> Virtual server IDs are not applicable to adapter-to-adapter mappings or OAuth use cases.

After pruning inapplicable policies and paths, the policy engine starts evaluating the request against the first applicable policy. Generally speaking, the policy engine moves on to the next applicable policy when it hits the end of an open-ended path, as indicated by an action of **Continue**, and stops when it hits the end of a closed-ended path, as indicated by an authentication policy contract or an action of **Done** or **Restart**. Depending on the policies, the policy engine might find an authentication source, a series of authentication sources, or no authentication source at all.

Default authentication sources

In the event that a request has only passed through an open-ended path and the policy engine finds no authentication source after evaluating the request through all the applicable policies, it picks the first applicable default authentication source. A default authentication source is considered applicable if it is mapped to the use case of the request.

If the policy engine cannot find a default authentication source and the **Fail if policy engine finds no authentication source** checkbox is cleared, PingFederate chooses an authentication source based on the following prioritized preferences:

1. If the request comes with an `IdpAdapterId` query parameter or a `pfidpaid` cookie, and if the authentication source specified by the query parameter or the cookie is mapped to the corresponding use case, PingFederate uses the specified authentication source. If the authentication source is not mapped, PingFederate denies the request and returns an error message.

   > **Note**
   >
   > If the request presents both the `IdpAdapterId` query parameter and the `pfidpaid` cookie, the `IdpAdapterId` query parameter takes precedence.

2. If the request comes with neither an `IdpAdapterId` query parameter nor a `pfidpaid` cookie, and if there is only one authentication source mapping, PingFederate uses the mapped authentication source.

   > **Note**
   >
   > If there are multiple authentication-source mappings, PingFederate returns the available authentication sources and lets the user authenticate through one of them. If the user selected the **Remember selection** checkbox and successfully authenticated, PingFederate returns a `pfidpaid` persistent cookie, identifying the user's preference.

   If the **Fail if policy engine finds no authentication source** checkbox is selected, PingFederate denies the request and returns an error message.

   > **Note**
   >
   > If a request has passed through a closed-ended path, the policy engine has already found at least one authentication source for the user. In this scenario, the policy engine ignores all default authentication sources.

Tracked HTTP request parameters

The policy engine is capable of tracking HTTP request parameters that it receives from the initial request and making them available to authentication sources, selector instances, and contract mappings throughout the policy.

Local identity profiles and authentication policy contracts

PingFederate empowers administrators to deliver empowers administrators to deliver a secure and easy-to-use customer authentication, registration, and profile management solution. A typical use case involves an HTML Form Adapter instance, a local identity profile, an authentication policy contract, and an IdP authentication policy. The HTML Form Adapter captures user attributes and maps them into an authentication policy contract through a local identity profile. In terms of configuration, the latter is accomplished by placing a local identity profile at the end of a policy path and completing the **Local Identity Mapping > Contract Fulfillment** configuration.

Policy enforcement based on OAuth clients

PingFederate can enforce authentication polices based on the requesting OAuth client, among other factors. To do that, you can include an OAuth Client Set Authentication Selector or an Extended Property Authentication Selector in a policy. Learn more in Configuring the OAuth Client Set Authentication Selector and Configuring the Extended Property Authentication Selector

Another option is to leverage the policy enforcement rule that PingFederate ignores authentication policy contract branches that are not mapped to an access token manager (ATM) that the requesting client can access. This rule also applies to policy branches that end on an authentication source. Learn more in Managing access token mappings.

Keep that policy enforcement rule in mind when you configure clients and ATMs. You can configure a client so that it can access only its default ATM. You can also configure an ATM so that only specific clients can access it. Those settings affect the authentication policy. Learn more in Configuring OAuth clients and Access token management.

Defining authentication policies

You can manage authentication policies and settings on the **Policies** page.

Steps

1. Go to **Authentication** > **Policies** > **Policies**.
2. On the **Policies** tab, select the **IdP Authentication Policies** checkbox to enable authentication policies for identity provider (IdP) browser single sign-on (SSO) requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.
3. Select the **SP Authentication Policies** checkbox to enable authentication policies for service provider (SP)-initiated browser SSO requests received at the `/sp/startSSO.ping` endpoint.

   > **Note**
   >
   > Selecting the **SP Authentication Policies** checkbox doesn’t enable authentication policies for IdP browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.

4. Select the **Fail if policy engine finds no authentication source** checkbox for PingFederate to deny the requests and return an error message when the policy engine finds no authentication source or authentication policy contract from the applicable policies and none of the default authentication sources are applicable. This checkbox is cleared by default.
5. On the **Policies** page, click **Add Policy** to create an authentication policy.

> **Tip**
>
> To create a new policy based on an existing policy, select **Copy** in the policy's **Select Action** menu.

1. On the **Policy** page, enter a name and, optionally, a description of the policy.
2. Select the **Handle Failures Locally** checkbox to keep users on the local IdP instead of returning them to the SP when an SP-initiated SSO request fails.

   When selected, after an SSO failure, the IdP uses the error template to give users more useful contextual information about the error than the SP can provide. The default error template is `idp.sso.error.page.template.html`.

   The checkbox is cleared by default.
3. In the **Policy** list, select an authentication source, IdP adapter instance or IdP connection, selector instance, or fragment.

   > **Note**
   >
   > If you started this new policy by copying an existing policy, your new policy is pre-populated. Modify the policy to suit your new use cases.

   > **Tip**
   >
   > When implementing your authentication requirements, think of authentication sources and selectors as checkpoints.

   Options

   For the PingID Adapter, IdP adapters developed using the `IdpAuthenticationAdapterV2` interface from the PingFederate SDK, including the HTML Form Adapter and SAML 2.0 IdP connections supporting the SP-initiated browser SSO profile, you can specify a user ID to be passed in from an earlier-factor adapter.

   Click **Options** and follow the on-screen instructions to select the source and the attribute to be used as the incoming user ID.

   > **Note**
   >
   > The **User ID Authenticated** checkbox must be selected for users to register as a new PingID user. Otherwise, the adapter automatically fails.

   Rules

   For any authentication source, you can optionally create one or more rules to define additional successful results. For example, if you want to deploy multi-factor authentication (MFA) using the PingID Adapter in stages by groups, you can create a rule to check for group membership information and only apply the PingID authentication flow to users who are members of certain groups.

   Click **Rules** and follow the on-screen instructions to manage your rules.

Copy and Paste

The **Copy** and **Paste** feature lets you copy a policy path and paste it into another place in the same policy, another policy, or a policy fragment. After you copy and paste a path, follow the on-screen instructions to correct any errors.

> **Tip**
>
> One benefit of this feature is that you can easily add a new step at the start or middle of an existing policy. To do that, copy and then remove the existing path below the point where you will define the new step. After you define the new step, paste the copied path back into the policy below the new step.

All results, including those based on rules, are displayed under the selected authentication source or selector instance. Each result forms a policy path.

4. For each policy path, select a policy action in the list.

> **Important**
>
> You can select **Fragments** as the policy action and, then, select a policy fragment that you have created. When you select a fragment, click **Fragment Mapping** and use the in-product help links to access the topics that describe how to configure the mapping. Learn about creating policy fragments in Policy fragments.

- If additional processing is required, repeat step 5c.
- If the policy path is extended from an authentication source and it is the end of the path, select **Done** or **Restart**, which marks this policy path as a closed-ended path.

  > **Tip**
  >
  > A policy path is closed-ended if it contains one or more authentication sources, with or without any selector instances. A closed-ended path can optionally end with an authentication policy contract or a local identity profile.

- If you need to reuse an authentication policy in multiple use cases, select an authentication policy contract or a local identity profile as the last policy action of a path, configure its contract fulfillment, and map the authentication policy contract to the applicable Browser SSO connections or OAuth grant-mapping configuration.

  Click **... Mapping** underneath your selection and then follow the on-screen instructions to complete the contract fulfillment configuration.

  > **Note**
  >
  > A policy path is also closed-ended if it ends with an instance of a custom authentication selector that returns an IdP adapter instance ID or the connection ID of an IdP connection. Because the custom selector returns an authentication source, such a closed-ended path cannot end with an authentication policy contract or a local identity profile. Instead, it must end with an action of **Done** or **Restart**.

The **Restart** policy action provides users the opportunity to do over. When triggered, the policy engine routes the requests back to the first checkpoint of the invoked authentication policy. It makes most sense to use the **Restart** policy action for a **Fail** policy path if the policy engine can route the request differently based on user input prompted by an authentication source. For a sample use case, refer to step 5 in Enabling third-party identity providers.

> **Note**
>
> Undesirable looping behaviors can occur if you select **Restart** for the **Fail** path at the root of an authentication policy tree. PingFederate mitigates this risk by automatically limiting the number of policy restarts per transaction.

- If the policy path is extended from a selector instance and it is the end of the path without any prior authentication source, select **Continue**, which leaves this path as an open-ended path.

> **Tip**
>
> A policy path is closed-ended if it contains one or more authentication sources, with or without any selector instances. A closed-ended path can optionally end with an authentication policy contract or a local identity profile.

5. Click **Done** to go back to the **Policies** page.

   Result:

   Your policy is enabled by default. As needed, toggle its status to disable the policy.

6. (Optional) Repeat step 5 to create additional authentication policies.

> **Important**
>
> The order of authentication policies matters because the policy engine starts from the first policy and works its way down. As needed, reorder your policies by using the up and down arrows.

7. If any individual policy is no longer required, select the **Delete** action or toggle its status to disable the policy.
8. (Optional) On the **Policies > Default Authentication Sources** tab, select one or more default authentication sources from the list for the policy engine to fall back on when it finds no authentication source from the applicable policies.

> **Important**
>
> Order matters because the policy engine starts from the first default authentication source on the list and works its way down. As needed, reorder your authentication sources by using the up and down arrows. There is no default selection.

9. (Optional) On the **Policies > Tracked HTTP Parameters** tab, add one or more HTTP request parameters to be tracked throughout a request.

> **Important**
>
> For each instance of the HTTP Request Parameter Authentication Selector that you place in a policy as the second, or subsequent, checkpoint, add its configured **HTTP Request Parameter Name** value here. By doing so, the policy engine preserves the parameter it receives from the initial request and makes it available to the selector instance throughout the policy. Learn more in Configuring the HTTP Request Parameter Authentication Selector.

10. Click **Save**.

Related links

- Managing policy contracts
- Managing local identity profiles

Specifying incoming user IDs

You can configure authentication policies to use incoming user identifiers at request time.

About this task

Some authentication sources make use of a user identifier at request time. For example:

- The PingID Adapter requires a user ID to be passed in from an earlier-authentication step to perform multi-factor authentication.
- The HTML Form Adapter and custom IdP adapters developed using the `IdpAuthenticationAdapterV2` interface from the PingFederate SDK can pre-populate username information based on an incoming user ID.
- A SAML 2.0 identity provider (IdP) connection can use an incoming ID to specify the `Subject` value in its authentication requests.
- An OpenID Connect IdP connection can leverage an incoming user ID to specify a `login_hint` parameter value in its OAuth authorization requests.

To address these use cases, specify the source and the attribute of the incoming user ID in an authentication policy.

You can select any IdP adapter instance or IdP connection that has been placed in the same policy path ahead of the current authentication source to be the source of the incoming user ID. After you select a source, choose an attribute from the selected IdP adapter contract or IdP connection. At runtime, the attribute value becomes the incoming user ID.

Alternatively, you can use the originating SAML 2.0, WS-Federation, or OpenID Connect authentication request as the source. In this scenario, the incoming user ID is derived from the `Subject` element in a SAML 2.0 authentication request, the `username` parameter in a WS-Federation authentication request, or the `login_hint` parameter in an OpenID Connect authentication request.

Steps

1. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the applicable authentication policy.
2. On the **Policy** tab, locate the authentication source that you need to provide an incoming user ID and then click **Options** underneath it.

3. On the **Incoming User ID** dialog, select the source of the incoming user ID from the **Source** list.

   > **Note**
   >
   > If you want the policy engine to derive the incoming user ID from the originating SAML 2.0 or WS-Federation authentication request, select **Context**.

4. Select an attribute of the incoming user ID from the **Attribute** list.

   > **Note**
   >
   > If you have selected **Context** in the previous step, select **Requested User** to derive the incoming user ID from the `Subject` element, the `username` parameter, or the `login_hint` parameter in the SAML 2.0, WS-Federation, or OpenID Connect authentication request, respectively.

   Result:

   During runtime, if a request does not originate from a SAML 2.0, WS-Federation, or OpenID Connect authentication request, or if the SAML 2.0, WS-Federation, or OpenID Connect authentication request does not include the optional `Subject` element, `username` parameter, or `login_hint` parameter, the policy engine advances without providing username information to the authentication source.

5. Select the **User ID Authenticated** check box to signal to the adapter that the value of the incoming user ID has been authenticated by a previous authentication source.

   Adapters can choose to enable certain functionality, such as registering new MFA devices, only when the user ID value has previously been authenticated.

   > **Warning**
   >
   > Don't enable **User ID Authenticated** for a self-service password reset authentication policy. Doing so can create a security vulnerability.

6. On the **Incoming User ID** dialog, click **Done**.

7. On the **Policy** tab, continue with the rest of your policy configuration.

Configuring rules in authentication policies

PingFederate supports more granular control through the use of rules in authentication policies.

About this task

An authentication source in an authentication policy has two results, Fail or Success, for which you can set one of the following actions:

- Append another authentication source for further processing.
- Append a selector for further processing.
- Select **Done** to terminate the authentication policy, making it a closed-ended path.
- Select an authentication policy contract or a local identity profile, also terminating the authentication policy, making it a closed-ended path.

Tip

A policy path is closed-ended if it contains one or more authentication sources, with or without any selector instances. A closed-ended path can optionally end with an authentication policy contract or a local identity profile.

Note

A policy path is also closed-ended if it ends with an instance of a custom authentication selector that returns an IdP adapter instance ID or the connection ID of an IdP connection. Because the custom selector returns an authentication source, such a closed-ended path cannot end with an authentication policy contract or a local identity profile. Instead, it must end with an action of **Done** or **Restart**.

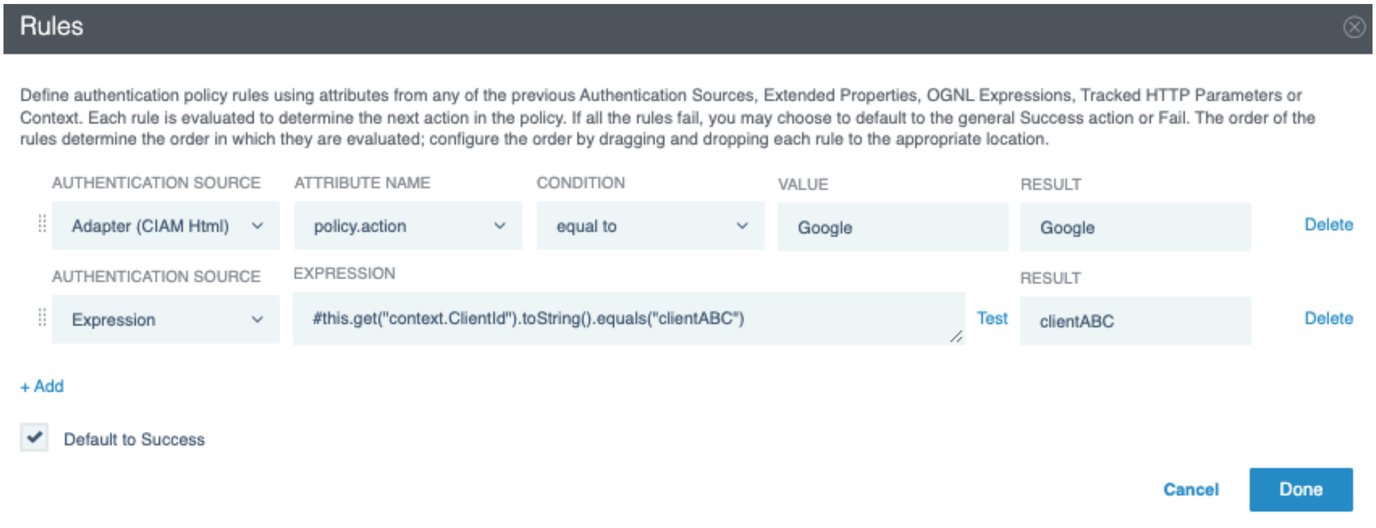
By applying multiple rules to an authentication source, an administrator can define additional, successful, results based on attribute values from the authentication source and set different action for each result.

For example, your OpenToken IdP Adapter instance returns an attribute, `EmployeeType`, that identifies the employee profile; a value of `temp` indicates the user is a contractor. Your organization mandates that all contractors must authenticate successfully against the OpenToken identity provider (IdP) adapter, followed by another IdP adapter, such as an instance of the PingID Adapter for multifactor authentication. To fulfill this authentication requirement, you can define a successful result by adding a rule to evaluate the `EmployeeType` value, and then select the PingID Adapter instance as the action for this match.

When multiple rules exist for a given authentication source, the first match wins. If no rule returns a match, administrators have the option to treat the authentication as successful or failure.

Steps

1. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the applicable authentication policy.
2. On the **Policy** tab, locate the authentication source that you want to define additional successful results for further processing, and then click **Rules** underneath it.
3. In the **Rules** window, select the source of the attribute from the **Authentication Source** list.



Rules

Define authentication policy rules using attributes from any of the previous Authentication Sources, Extended Properties, OGNL Expressions, Tracked HTTP Parameters or Context. Each rule is evaluated to determine the next action in the policy. If all the rules fail, you may choose to default to the general Success action or Fail. The order of the rules determine the order in which they are evaluated; configure the order by dragging and dropping each rule to the appropriate location.

AUTHENTICATION SOURCE	ATTRIBUTE NAME	CONDITION	VALUE	RESULT	
Adapter (CIAM Html)	policy.action	equal to	Google	Google	Delete

AUTHENTICATION SOURCE	EXPRESSION		RESULT	
Expression	#this.get("context.ClientId").toString().equals("clientABC")	Test	clientABC	Delete

+ Add

☑ Default to Success

4. Select an attribute from the **Attribute Name** list.

The rule can use any attribute processed in a previous step of the policy.

Authentication Sourceoptions	Attribute Nameoptions
Previous authentication sources	The attribute options depend on the previous authentication source.
Extended Properties (available only if PingFederate has extended properties configured)	The attribute options depend on which extended properties were configured.
Tracked HTTP Parameters (available only if PingFederate has tracked HTTP parameters configured)	The attribute options depend on which tracked HTTP parameters were configured.
Context	◦ **Client ID** for OAuth flows ◦ **Scope** for OAuth flows ◦ **SP Connection Entity ID** for SAML flows ◦ **Virtual Server ID** for SAML flows ◦ **HTTP Request** ◦ **Client IP**
Expression	When you select **Expression** as the **Authentication Source**, enter an OGNL expression in the field. The expression can reference upstream attributes, extended properties, tracked parameters, and context. When the expression's boolean result is true, PingFederate returns the rule's **Result** value. When the expression's boolean result is false, PingFederate evaluates the next rule. To test an expression rule, click **Test**. Then use the **Test Expression** dialog to test the expression with different values. **Caution** Policy rules that have overly restrictive expressions or expressions with wildcards might result in denial-of-service or unauthorized access.

5. From the **Condition** list, select how PingFederate should compare the value that you are going to specify in the next step against the attribute value from the authentication source.

   The choices are:

   - equal to
   - equal to (case insensitive)
   - equal to DN
   - not equal to

- not equal to (case insensitive)
- not equal to DN
- multi-value contains
- multi-value contains (case insensitive)
- multi-value contains DN
- multi-value does not contain
- multi-value does not contain (case insensitive)
- multi-value does not contain DN

Use one of the first six choices only for attributes consisting of a single value. Use the multi-value conditions when you want PingFederate to verify whether an attribute contains or does not contain the specified value in its attribute list.

> **Caution**
>
> Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

6. In the **Value** field, enter the desired value to be compared against the attribute value from the authentication source.
7. In the **Result** field, enter a unique label.
8. **Optional:** To add another rule, click **Add** and repeat steps step 3 to step 7.

> **Tip**
>
> If there are multiple rules, you can use the drag handle icon⠿ to change the order of the rules. PingFederate evaluates the rules in the order you place them.

9. Select the **Default to Success** check box if you want the policy engine to treat the authentication attempt as successful when no rules return a match.

> **Note**
>
> The **Default to Success** check box is selected by default. When you clear this check box, the policy engine treats the attempt as a failure when no rules return a match.

10. To close the **Rules** window, click **Done**.

Result:

Your policy is now updated with a new policy path, or paths if you have added multiple rules.

For instance, if you have added two rules with labels **Contractors**, the first rule, and **Senior executives**, the second rule, to an authentication source, you should see the following results in the policy:

- **Fail**

  - **Contractors**, a new result based on the first rule
  - **Senior executives**, a new result based on the second rule
  - **Success**, available only when the **Default to Success** check box is selected

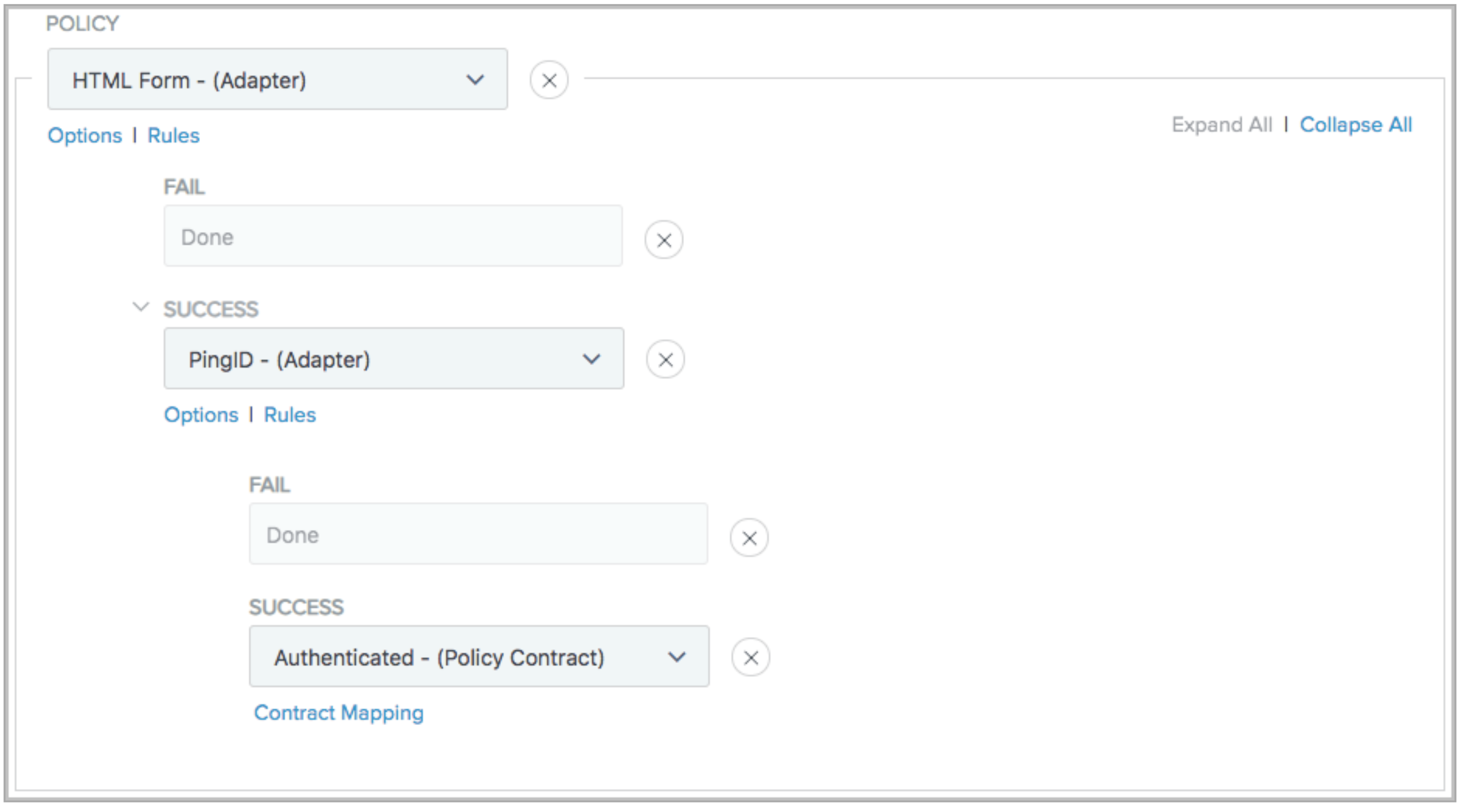
11. On the **Policy** window, continue with the rest of your policy configuration.

Defining authentication policies based on group membership information

PingFederate lets you configure authentication policies based on group membership information through the use of rules.

About this task

Assume you have created the following authentication policy to enforce multifactor authentication using PingID after the users have successfully authenticated against an HTML Form Adapter instance.



While this policy satisfies the authentication requirements, you might prefer to roll out multi-factor authentication based on group membership over a period of time. To accomplish this policy deployment strategy, you can use rules to define the applicable groups and set different policy actions accordingly.

As an example, suppose you want to enforce PingID multi-factor authentication to two Active Directory (AD) groups:

- CN=helpdesk,OU=IT,DC=example,DC=com (IT helpdesk personnel)
- CN=leads,OU=IT,DC=example,DC=com (Leaders in the IT department)

Steps

1. If you have not done so, create a new AD group, for example, `CN=PingIDRequired,OU=IT,DC=example,DC=com`, and place those two groups as members of the new group.

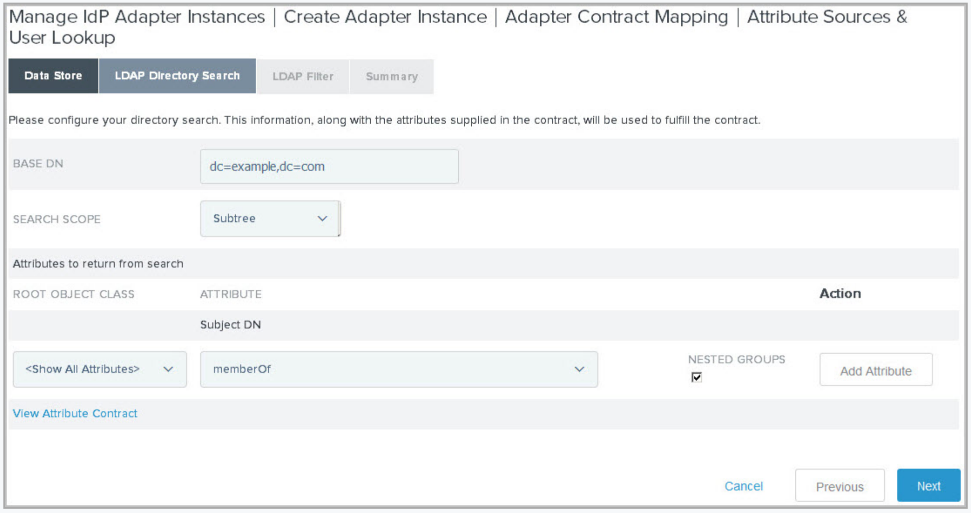
   > **Tip**
   >
   > Generally speaking, this step streamlines the process of deploying your authentication policies to additional groups of users later. In other words, when you are ready to roll out your authentication policies to more users, simply add the applicable groups as members of the new group. This way, you are not required to make any changes to the authentication policies, after they are configured.

2. If you have not done so, follow these steps to extend the HTML Form Adapter instance to return group membership information from your AD.

   1. Extend the HTML Form Adapter instance with the `memberOf` attribute on the **Extended Contract** window.

   2. Configure PingFederate to fulfill the `memberOf` attribute from your AD.

      > **Note**
      >
      > Because the actual groups are nested inside the new group created in step 2, configure the IdP adapter contract to pull the `memberOf` attribute values with nested groups on the LDAP Directory Search window. For more information, see Defining the IdP adapter contract.



Manage IdP Adapter Instances | Create Adapter Instance | Adapter Contract Mapping | Attribute Sources & User Lookup

Please configure your directory search. This information, along with the attributes supplied in the contract, will be used to fulfill the contract.

BASE DN: dc=example,dc=com

SEARCH SCOPE: Subtree

Attributes to return from search

ROOT OBJECT CLASS	ATTRIBUTE		Action
	Subject DN		
<Show All Attributes>	memberOf	NESTED GROUPS ☑	Add Attribute

3. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the applicable authentication policy.

4. On the **Policy** window, click **Rules** underneath the HTML Form Adapter instance.

5. In the **Rules** dialog, add a rule to check for membership information obtained from the HTML Form Adapter instance.

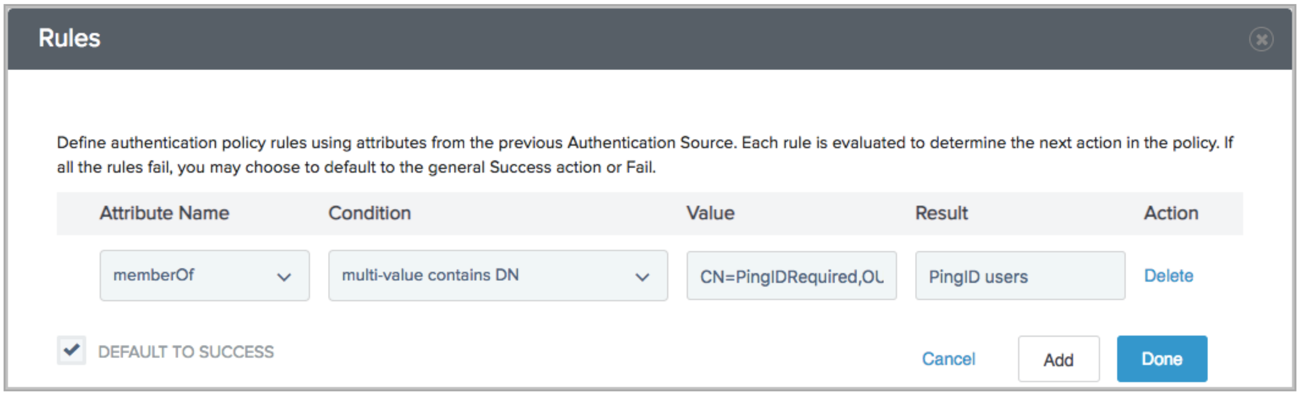
   1. From the **Attribute Name** list, select `memberOf`.

2. Select how PingFederate should compare the value that you are going to specify in the next step against the attribute value from the HTML Form Adapter instance. For example, **multi-value contains DN** works well for the `memberOf` AD user attribute.
3. Enter the distinguished name (DN) of the new group created in step 2 in the **Value** field, such as `CN=PingIDRequired,OU=IT,DC=example,DC=com`.
4. In the **Result** field, enter a label, for example, `PingID users`.
5. Leave the **Default to Success** check box as selected.

> **Note**
>
> When the **Default to Success** check box is selected, you can set the action for the scenario where none of the rules returns a match.

Example:

Your **Rules** dialog should be similar to the following sample.



Rules

Define authentication policy rules using attributes from the previous Authentication Source. Each rule is evaluated to determine the next action in the policy. If all the rules fail, you may choose to default to the general Success action or Fail.

Attribute Name	Condition	Value	Result	Action
memberOf	multi-value contains DN	CN=PingIDRequired,OL	PingID users	Delete

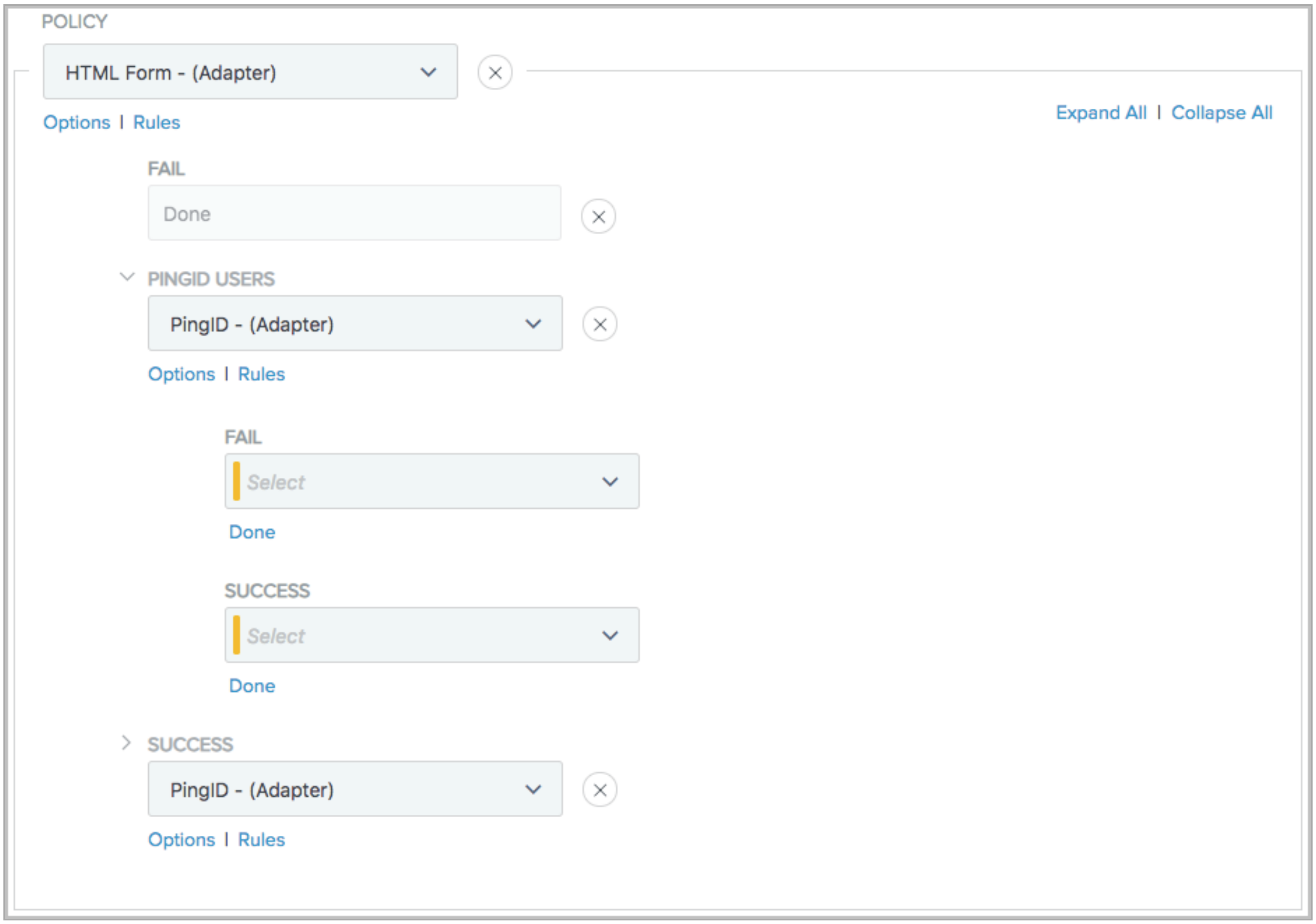
☑ DEFAULT TO SUCCESS

6. To close the **Rules** window, click **Done**.
7. For the new **HTML Form > PingID users** policy path, select your PingID Adapter instance as the policy action from the list.

> **Note**
>
> To open the **Incoming User ID** dialog, click **Options**, underneath the PingID Adapter instance. Configure the source of the user ID required by the PingID Adapter, and then click **Done** to close the dialog. For more information, see Specifying incoming user IDs.

Result:

Your policy should be similar to the following sample.



There are four policy paths:

- **HTML Form > Fail**
- **HTML Form > PingID users > Fail**
- **HTML Form > PingID users > Success**
- **HTML Form > Success**

8. Update your policy as follows:

HTML Form > Fail

Leave **Done** as the policy action. At runtime, PingFederate terminates the request and returns an error message to the user.

HTML Form > PingID users > Fail

Select **Done** as the policy action. At runtime, PingFederate terminates the request and returns an error message to the user.

HTML Form > PingID users > Success

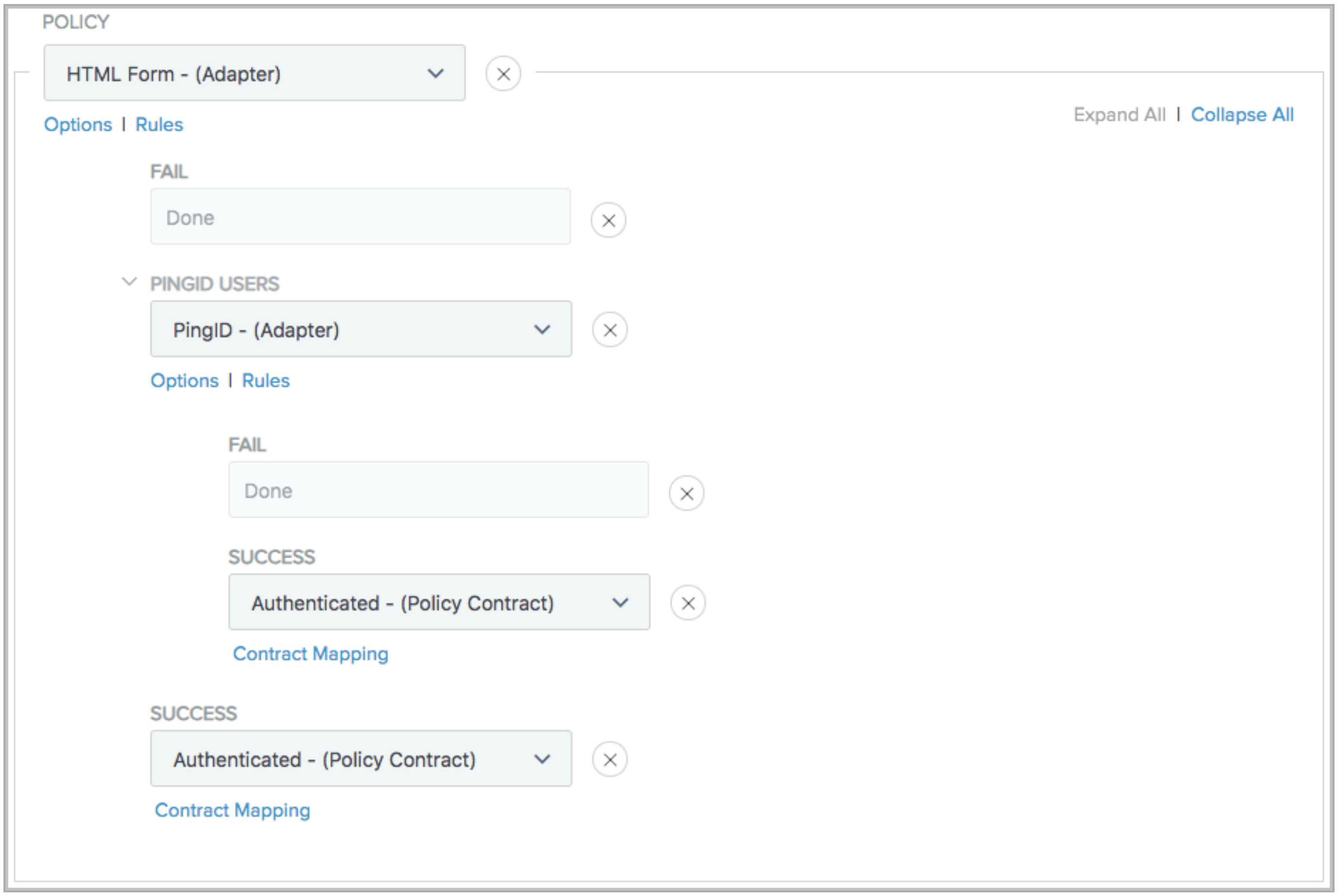
Select an authentication policy contract as the policy action. Click **Contract Mapping** to complete its fulfillment. At runtime, PingFederate fulfills the authentication policy contract and carries on with the request.

HTML Form > Success

Reconfigure the policy action for this policy path. Select an authentication policy contract as the policy action. Click **Contract Mapping** to complete its fulfillment. At runtime, PingFederate fulfills the authentication policy contract and carries on with the request.

Result:

Your policy should be similar to the following sample.



POLICY

HTML Form - (Adapter)

Options | Rules

Expand All | Collapse All

FAIL

Done

PINGID USERS

PingID - (Adapter)

Options | Rules

FAIL

Done

SUCCESS

Authenticated - (Policy Contract)

Contract Mapping

SUCCESS

Authenticated - (Policy Contract)

Contract Mapping

9. To close the **Policy** window, click **Done**.

10. On the **Policies** window, click **Save**.

> **Note**
>
> Group membership is only one of the possible factors that you can use to define additional policy paths and their policy actions. In general, you can use any attributes available from the authentication source when configuring rules.

Applying policy contracts or identity profiles to authentication policies

To apply an authentication policy contract to a policy, select an authentication policy contract or a local identity profile as the last action of one or more closed-ended paths and configure fulfillment for each contract.

About this task

An authentication policy contract can harness attribute values obtained from all authentication sources along the path leading up to it. Administrators can select the same authentication policy contract or local identity profile for different closed-ended paths, in one or more authentication policies, and fulfill them differently to suit the requirements. To enforce the same set of authentication policies in multiple use cases, map the authentication policy contract to the applicable Browser SSO connections and OAuth grant-mapping configuration.

Steps

1. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the applicable authentication policy.
2. On the **Policy** window, locate all closed-ended paths in the policy.

   A policy path is closed-ended if it contains one or more authentication sources, with or without any selector instances. A closed-ended path can optionally end with an authentication policy contract or a local identity profile.

+

> **Note**
>
> A policy path is also closed-ended if it ends with an instance of a custom authentication selector that returns an IdP adapter instance ID or the connection ID of an IdP connection. Because the custom selector returns an authentication source, such a closed-ended path cannot end with an authentication policy contract or a local identity profile. Instead, it must end with an action of **Done** or **Restart**.

+ Consider the following sample policy.

+ image::wko1564003311406.png[alt="A screen capture illustrating a sample policy with four closed-ended paths and one open-ended path",role="border-no-padding"]

+ This policy has two selector instances, **Test** and **Retail**, two identity provider (IdP) adapter instances, and five policy paths:

- **Test** > **No** > **HTML Form** > **Fail**
- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **No**
- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **Yes** > **PingID** > **Fail**
- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **Yes** > **PingID** > **Success**
- **Test** > **Yes**

  The first four paths are closed-ended while the last path is open-ended.

  1. Select **Done** as the policy action for the following paths:

- **Test** > **No** > **HTML Form** > **Fail**
- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **Yes** > **PingID** > **Fail**

Result:

At runtime, PingFederate terminates the request and returns an error message to the user.

1. Select the applicable authentication policy contract or local identity profile as the policy action for the rest of the closed-ended paths:

- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **No**
- **Test** > **No** > **HTML Form** > **Success** > **Retail** > **Yes** > **PingID** > **Success**

Suppose your use case does not involve consumer authentication, registration, and profile management. It makes sense to select an authentication policy contract for the **PingID > Success** result, because the users have successfully met all your authentication requirements.

At runtime, PingFederate fulfills the authentication policy contract and carries on with the request.

Depending on your use case, you might also select an authentication policy contract for the **PingID > Fail** result, possibly with an attribute indicating that the users have failed a certain part of your authentication requirements, and make other authorization decision using the Token Authorization framework in the applicable connections later.

1. For each selected authentication policy contract, if any, click **Contract Mapping** and then complete the **Manage Authentication Policies > Authentication Policy Contract Mapping** workflow to complete the configuration. For more information, see Configuring contract mapping.
2. For each selected local identity profile, if any, click **Local Identity Mapping** and then complete the **Manage Authentication Policies > Inbound Mapping & Contract Fulfillment** workflow to complete the configuration. For more information, see Configuring local identity mapping.
3. Select **Continue** as the policy action for the open-ended path **Test > Yes**.

   Result:

At runtime, PingFederate skips to the next policy. Your policy should be similar to the following sample.

+ image::ssb1564003312263.png[alt="A screen capture illustrating a sample policy with four closed-ended paths and one open-ended path",role="border-no-padding"]

1. To close the **Policy** window, click **Done**.
2. On the **Policies** window, click **Save**.

Related links

- Managing local identity profiles
- Managing policy contracts

Configuring contract mapping

For each authentication policy contract mapping, configure the sources of its attributes and specify any criteria for issuing the contract.

Steps

1. Go to **Authentication** > **OAuth** > **Authentication Policy Contract Mapping**.

2. On the **Authentication Policy Contract** list, click the desired mapping or select the desired mapping from the **Authentication Policy Contract** list.

   > **Note**
   >
   > If you don't already have an authentication policy contract mapping configured, go to **Authentication > Policies > Policy Contracts** to configure and save a new contract.

3. **Optional:** On the **Attribute Sources & User Lookup** window, click **Add Attribute Source** to configure datastore queries.

4. On the **Contract Fulfillment** tab, fulfill the selected contract.

   > **Note**
   >
   > If the selected closed-ended path contains more than one authentication source, you have access to attributes obtained successfully from the previous authentication sources along the same path.
   > For example, referring to the earlier policy in Applying policy contracts or identity profiles to authentication policies, if you select an authentication policy contract for the **PingID (Adapter) > Success** result, you can map attributes from the HTML Form Adapter and the PingID Adapter.
   > Besides the preceding identity provider (IdP) connection or IdP adapter instance, you can also use the following as the source of fulfillment:
   >
   > - Dynamic text
   > - Attribute mapping expressions, if enabled
   > - Tracked HTTP request parameters, if configured
   > - Request context
   > - Extended properties, if configured on the **Extended Properties** window

5. **Optional:** On the **Issuance Criteria** tab, configure conditions to be validated before issuing an authentication policy contract.

   For more information, see Defining issuance criteria for contract or local identity mapping.

6. On the **Summary** tab, review your configuration, modify as needed, and then click **Done**.

7. On the **Policy** window, continue with the rest of your policy configuration.

Configuring local identity mapping

You can configure your local identity mapping in the PingFederate administrative console.

Steps

1. On the **Inbound Mapping** tab, configure the attribute mappings for registration and profile management.

   > **Note**
   >
   > At runtime, PingFederate fulfills the value of the `pf.local.identity.unique.id` built-in local identity field based on this configuration and passes the value to PingDirectory. The `pf.local.identity.unique.id` field value should therefore be mapped from the subject identifier of the preceding authentication source. You can also map other local identity fields so that PingFederate can streamline the registration process by pre-populating values on the registration page.

> **Important**
>
> This configuration overrides the default field values configured within the local identity profile. For more information, see Configure a local identity field.

This tab does not apply and stays hidden if your use case does not involve registration and profile management. See Enabling third-party identity providers without registration.

2. **Optional:** On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source** to configure datastore queries.

3. On the **Contract Fulfillment** tab, fulfill the authentication policy contract associated with the selected local identity profile.

   If the selected closed-ended path contains more than one authentication source, you have access to attributes obtained successfully from the previous authentication sources along the same path.

   For example, select your local identity profile in the **Source** column and the desired local identity field in the **Value** column.

> **Note**
>
> If your use case doesn't involve registration or profile management, the source of fulfillment is limited to:
>
> - The preceding identity provider (IdP) connection or IdP adapter instance
> - Dynamic text
> - Attribute mapping expressions, if enabled
> - Tracked HTTP request parameters, if configured
> - Request context
> - Extended properties, if configured on the **Extended Properties** window

4. **Optional:** On the **Issuance Criteria** tab, configure conditions to be validated before issuing an authentication policy contract.

   For more information, see Defining issuance criteria for contract or local identity mapping.

5. On the **Summary** tab, review your configuration, modify as needed, and then click **Done**.

6. On the **Policy** window, continue with the rest of your policy configuration.

Defining issuance criteria for contract or local identity mapping

You must define certain criteria for contract or local identity mapping in PingFederate to process a request.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

2. In the **Attribute Name** list, select the attribute to be evaluated.

3. In the **Condition** list, select the comparison method.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**
   - **not equal to**
   - **not equal to (case insensitive)**
   - **not equal to DN**
   - **multi-value contains**
   - **multi-value contains (case insensitive)**
   - **multi-value contains DN**
   - **multi-value does not contain**
   - **multi-value does not contain (case insensitive)**
   - **multi-value does not contain DN**

   > **Note**
   >
   > The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

4. In the **Value** field, enter the comparison value.

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

5. In the **Error Result** field, enter a custom error message.

   To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

   If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.

7. **Optional:** Repeat to add more criteria.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.

   2. In the **Expression** field, enter the required expressions.

   3. **Optional:** In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.

   5. **Optional:** Click **Test**, enter values in the applicable fields, and verify the results.

   6. **Optional:** Repeat to add multiple criteria using attribute mapping expressions.

Related links

- Token authorization
- Localizing messages for end users
- Attribute mapping expressions

Mapping a policy contract to multiple use cases

The last step to reuse an authentication policy in multiple service provider (SP) connections is to map the authentication policy contract into the applicable SP connections.

About this task

In general, for identity provider (IdP) browser single sign-on (SSO) use cases, if you select authentication policy contracts in your authentication policies then you must map the authentication policy contracts to the applicable SP connections.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.
2. Select the applicable SP connection from the list of connections.
3. On the **Activation & Summary** tab, click **Authentication Source Mapping**.
4. Click **Map New Authentication Policy** and use the in-product help on each screen as needed to map the authentication policy contract into the SP connection.

Result

Similarly, to reuse an authentication policy for browser-based OAuth authorization code and implicit flows, map the authentication policy contract to the applicable browser SSO connections and OAuth grant-mapping configuration. For more information, see Managing authentication policy contract grant mapping.

SP authentication policies

Service provider (SP) authentication policies provide a means for you to impose authentication requirements on SP-initiated browser single sign-on (SSO) requests received at the `/sp/startSSO.ping` endpoint.

When you enable this optional feature, you create policies that the PingFederate SP server can use to find the applicable SP adapter instance to access target applications. For this reason, you must configure the target applications to provide the `SpSessionAuthnAdapterId` parameter or the `TargetResource` parameter, or both, in their SP-initiated SSO requests.

If you prefer to provide the `TargetResource` parameter without the `SpSessionAuthnAdapterId` parameter, you must go to **Applications** > **Integration** > **Target URL Mapping** and configure entries to map the `TargetResource` values to the applicable SP adapter instances.

> **Note**
>
> SP authentication policies only apply to SP-initiated browser SSO requests received at the #spStartSsoPing SP application endpoint. They do not apply to unsolicited SSO requests received at the SP protocol endpoints.
> In addition, enabling SP authentication policies does not enable authentication policies for identity provider (IdP) browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.

For more information and configuration steps, see the subsequent sample use cases.

Configuring an SP authentication policy for users from one IdP

You can configure a service provider (SP) authentication policy to enforce authentication requirements for an identity provider (IdP) connection.

Before you begin

This example requires the following components:

- An SP adapter instance deployed, configured, and integrated with the target application.

- An IdP connection to the partner. For more information, seestep 1.
- An IdP connection to the third-party IdP that facilitates the multifactor authentication process. For more information, see step 2.
- An authentication policy contract to carry user attributes from the partner to the target application. For more information, see step 3.
- An SP authentication policy. For more information, see step 4 and step 7.
- An adapter mapping between the authentication policy contract and the applicable SP adapter instance. For more information, see step 5.
- An SP-initiated single sign-on (SSO) URL. For more information, see step 6.

About this task

In this example, you want to create an IdP connection to Alpha, which passes two attributes in its assertions, `SAML_SUBJECT` and `samlEmail`, on your PingFederate SP server. You also want to enforce multi-factor authentication (MFA) for users from Alpha through Bravo, a third-party IdP that returns only the `SAML_SUBJECT` attribute and requires a user ID to be passed in from the original source. Both Alpha and Bravo support SAML 2.0 and only the SP-initiated single sign-on (SSO) profile.

Create an SP adapter instance using the **Applications > Integration > SP Adapters** configuration wizard and complete the last-mile integration with the target application. The SP adapter instance name and ID are **Sample** and `sample`, respectively. On **System > Server > Protocol Settings > Federation Info**, the base URL for your PingFederate SP server is `https://sso.xray.local:9031`. There are no other IdP connections besides those required to connect with Alpha and Bravo.

Steps

1. Go to **Authentication > Integration > IdP Connections**. On the **IdP Connections** window, click **Create Connection**.

    1. Follow the connection workflow to create a SAML 2.0 IdP connection to Alpha.

        In this example, Alpha's entity ID is `sso.alpha.local`.

    2. On the **SAML Profiles** tab, make sure that the **SP-Initiated SSO** check box is selected.
    3. On the **Identity Mapping** window, select **No Mapping**.

        > **Tip**
        >
        > If the partner and your organization agree to support account linking, select **Account Linking**. If the partner and your organization agree to support the IdP-initiated profile, select **Account Mapping** or **Account Linking**.
        > Both use cases require the applicable SP adapter instance (**Sample**) to be mapped into the connection. To get to the **Target Session Mapping** tab, go to **IdP Connection > Browser SSO > User-Session Creation**. The rest of the steps remain unchanged.
        > This sample configuration uses neither **Account Linking** or **Account Mapping**.

    4. Complete the rest of the connection configuration.

2. Repeat step 1 to create a SAML 2.0 IdP connection to Bravo.

    In this example, Bravo's entity ID is `sso.bravo.local`.

3. Go to **Authentication** > **Policies** > **Policy Contracts**. On the **Policy Contracts** window, click **Create New Contract**.

> **Tip**
>
> The purpose of an authentication policy contract is to harness user attributes obtained through one or more authentication sources as the request flows through the applicable authentication policy. It is the medium between the authentication policies and the target applications. In general:
>
> - You map attributes to authentication policy contracts from authentication policies. For more information, see step 4 in this example.
> - You map attributes from authentication policy contracts to target applications through adapter mappings. For more information, see step 5 in this example.

   1. On the **Contract Info** tab, enter a name for this authentication policy contract.

In this example, the name of the policy contract is `Authenticated`.

   1. On the **Contract Attributes** tab, enter `mail` under **Extend the Contract** and click **Add**.

   2. Complete the rest of the connection configuration.

      Result:

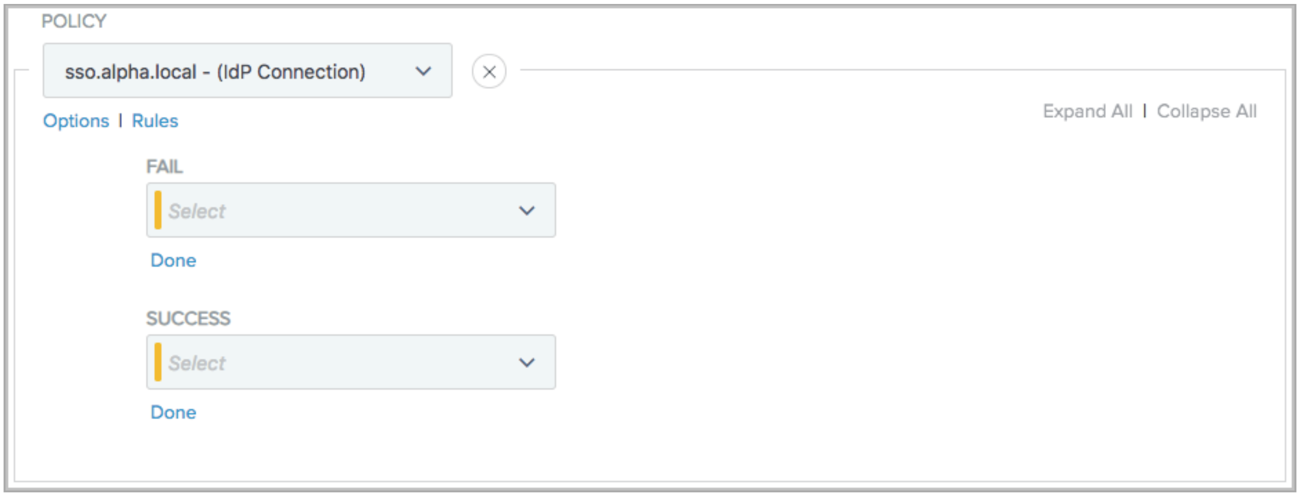
      When finished, your policy contract has two attributes: `subject` and `mail`.

4. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, click **Add Policy** to define a policy to enforce the third-party authentication requirement.

   1. On the **Policy** window, enter a name and a description for this policy, and select **sso.alpha.local (IdP Connection)** as the first policy action from the list.

      Result:

      An IdP connection has two results, **Fail** and **Success**, as illustrated in the following screen capture.



      Each result forms its own policy path that requires further configuration.

2. For the **sso.alpha.local** > **Fail** path, click **Done** as the policy action.

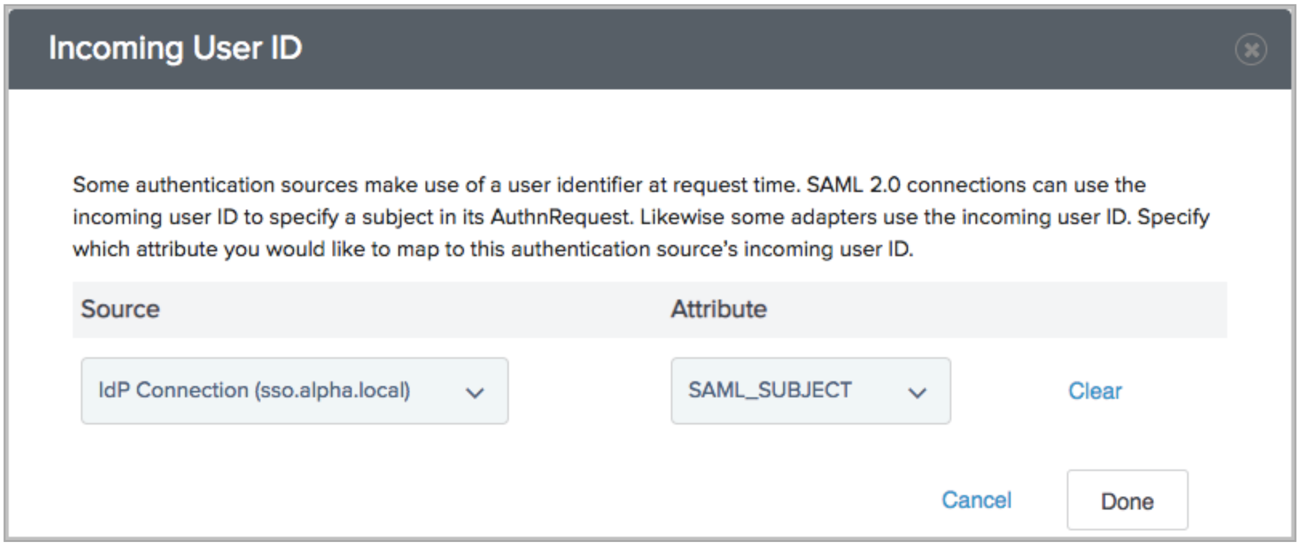
   Result:

   At runtime, PingFederate terminates the request and returns an error message to the user.

3. From the **sso.alpha.local** > **Success** list, select **sso.bravo.local** as the policy action.

4. Below **sso.bravo.local**, click **Options** to relay the user ID (`SAML_SUBJECT`) from `sso.alpha.local` to `sso.bravo.local` .

5. On the**Incoming User ID** dialog window, select **IdP Connection (sso.alpha.local)** from the **Source** list and **SAML_SUBJECT** from the **Attribute** list.

   Incoming User ID

   Some authentication sources make use of a user identifier at request time. SAML 2.0 connections can use the incoming user ID to specify a subject in its AuthnRequest. Likewise some adapters use the incoming user ID. Specify which attribute you would like to map to this authentication source's incoming user ID.

Source	Attribute	
IdP Connection (sso.alpha.local)	SAML_SUBJECT	Clear

   To close the **Incoming User ID** dialog window, click **Done**.

6. For the **sso.bravo.local** > **Fail** path, select **Done** as the policy action.

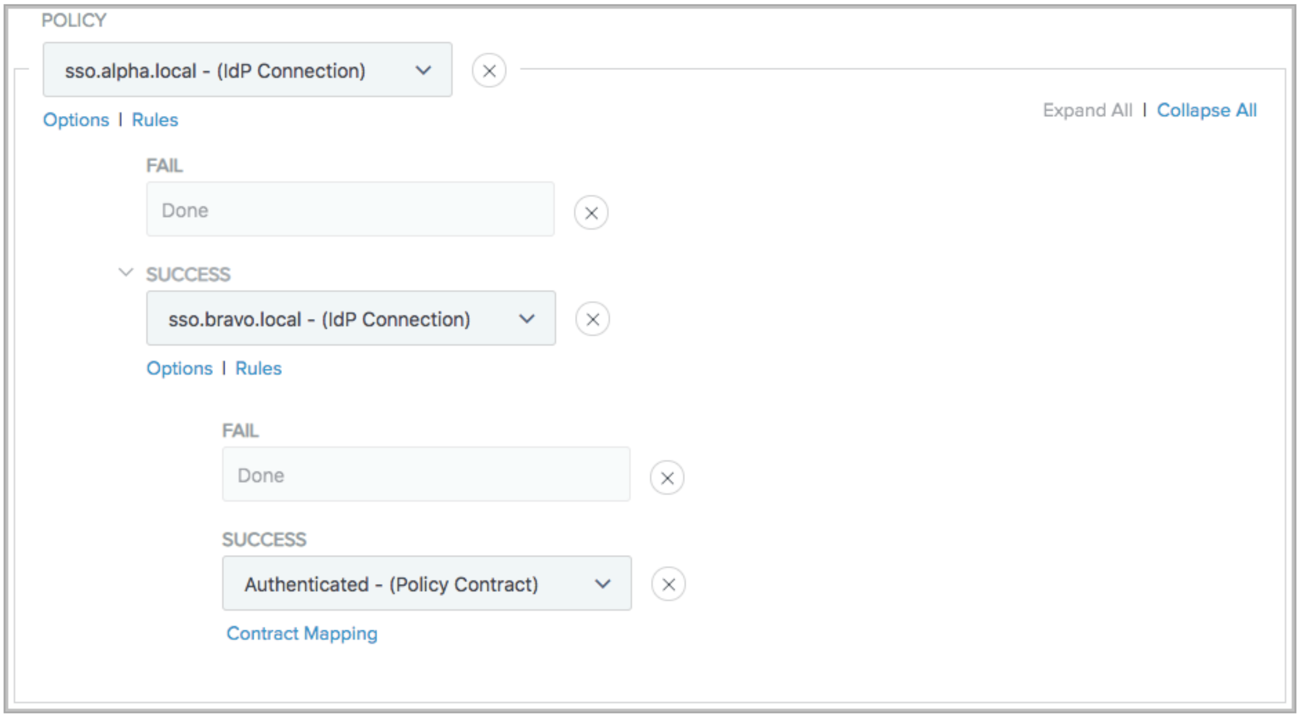
   Result:

   At runtime, PingFederate terminates the request and returns an error message to the user.

7. For the **sso.bravo.local** > **Success** path, select the authentication policy contract created in step 3.

   Result:

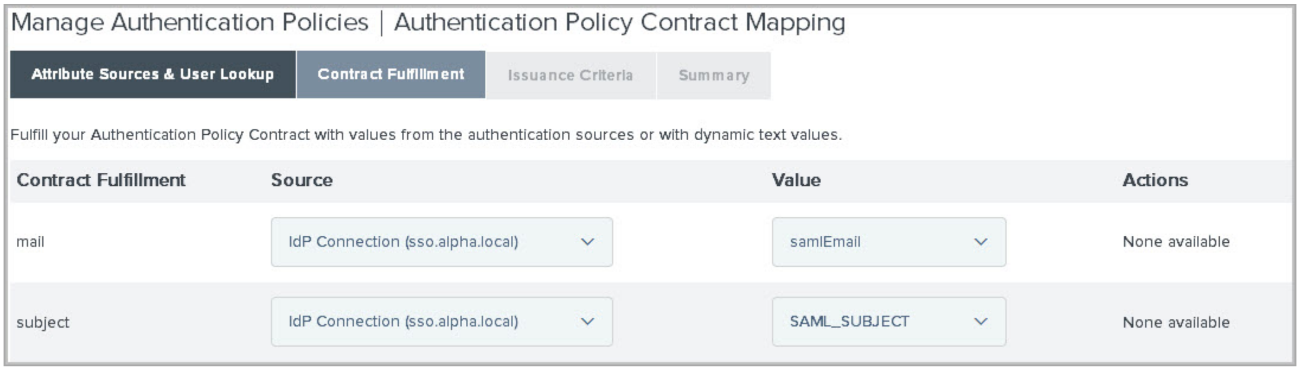
   Your policy should be similar to the following sample.


POLICY
sso.alpha.local - (IdP Connection)
Options | Rules
Expand All | Collapse All
FAIL
Done
SUCCESS
sso.bravo.local - (IdP Connection)
Options | Rules
FAIL
Done
SUCCESS
Authenticated - (Policy Contract)
Contract Mapping

8. Below the authentication policy contract, click **Contract Mapping** and follow the **Manage Authentication Policies > Authentication Policy Contract Mapping** configuration workflow to configure the fulfillment of the authentication policy contract.

9. On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source** to configure datastore queries.

10. On the **Contract Fulfillment** tab, from the **Source** list select **IdP Connection (sso.alpha.local)** and the appropriate attributes from the **Value** list to fulfill the authentication policy contract attributes.


Manage Authentication Policies | Authentication Policy Contract Mapping
Attribute Sources & User Lookup
Contract Fulfillment
Issuance Criteria
Summary

Fulfill your Authentication Policy Contract with values from the authentication sources or with dynamic text values.

Contract Fulfillment	Source	Value	Actions
mail	IdP Connection (sso.alpha.local)	samlEmail	None available
subject	IdP Connection (sso.alpha.local)	SAML_SUBJECT	None available

Tip

In this configuration, you are mapping attributes to an authentication policy contract from the authentication policy.

```
[.b]**Optional:** On the [.uicontrol]**Issuance Criteria** tab, configure conditions to be
validated before issuing an authentication policy contract.
```

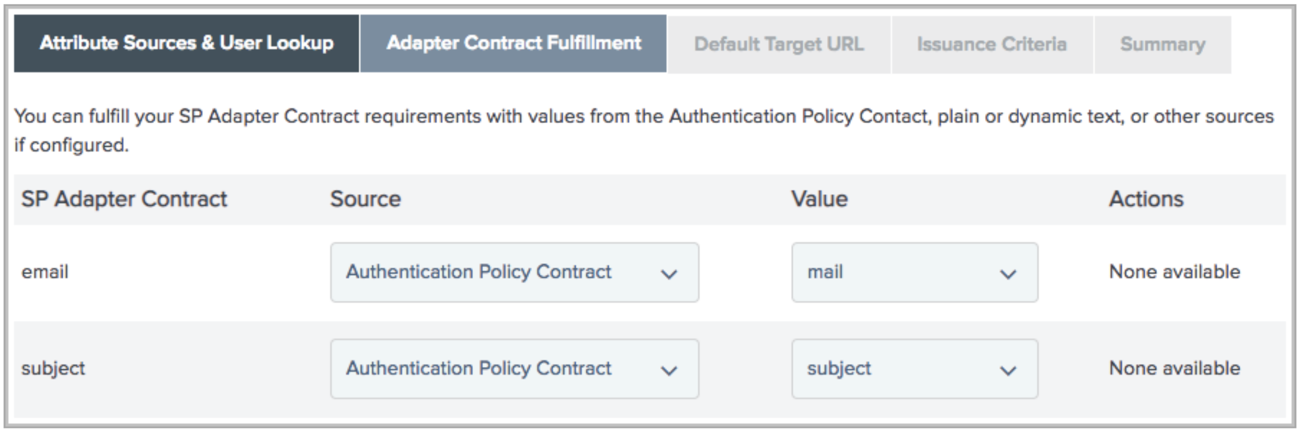
On the **Summary** tab, click **Done**.

On the **Policy** window, click **Done**.

On the **Policies** window, click **Save**.

5. Go to **Applications** > **Integration** > **Policy Contract Adapter Mappings**. Map the authentication policy contract to the SP adapter instance that you have deployed, configured, and integrated with the target application.

    1. From the **Source Instance**list, select the authentication policy contract, created in step 3.
    2. From the **Target Instance** list, select the applicable SP adapter instance (**Sample**).
    3. Click **Add Mapping**.
    4. Follow the **Mapping Configuration** workflow to create the mapping.
    5. On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source** to configure datastore queries.
    6. On the **Adapter Contract Fulfillment** window, select **Authentication Policy Contract** from the **Source** list and the appropriate contract attributes from the **Value** list to fulfill the SP adapter contract.



You can fulfill your SP Adapter Contract requirements with values from the Authentication Policy Contact, plain or dynamic text, or other sources if configured.

SP Adapter Contract	Source	Value	Actions
email	Authentication Policy Contract	mail	None available
subject	Authentication Policy Contract	subject	None available

> **Tip**
>
> In this configuration, you are mapping attribute values from the authentication policy contract to the target application through the applicable SP adapter instance.

```
[.b]**Optional:** On the [.uicontrol]**Default Target URL** tab, specify a default target URL for
this mapping configuration.
[.b]**Optional:** On the [.uicontrol]**Issuance Criteria** tab, configure conditions to be
validated before issuing an SP adapter contract.
```

Click **Done**.

6. Configure an SP-initiated SSO URL in your target application by combining the base URL of your PingFederate SP server, `https://sso.xray.local:9031`, the PingFederate's application SSO endpoint, `/sp/startSSO.ping`, and the `SpSessionAuthnAdapterId` parameter with the adapter ID of the applicable SP adapter instance as the parameter value.

    For example: https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sample

    If you have not defined a default URL for the adapter mapping, configured in step 5, the IdP connection, or the PingFederate SP server, you must also configure your target application to include the `TargetResource` parameter in its SP-initiated SSO requests.

> **Important**
>
> When using the parameters `TargetResource` or `TARGET` with their own query parameters included, the parameter value must be URL-encoded. Any other parameters that contain restricted characters, such as many SAML URNs, also must be URL-encoded. For information about URL encoding, see third party resources such as HTML URL-encoding Reference. Parameters are case-sensitive.

7. When you are ready to test your new use case, go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the **SP Authentication Policies** check box to enable policies for SP-initiated Browser SSO requests received by at the `/sp/startSSO.ping` endpoint, and click **Save**.

Configuring SP authentication policies for users from multiple IdPs

You can configure service provider (SP) authentication policies to handle different authentication requirements for multiple identity provider (IdP) connections.

About this task

Assume you configure the following use cases in an earlier version of PingFederate:

- Two SP adapter instances on **Applications** > **Integration** > **SP Adapters**.

Instance Name	Instance ID	Extended Contract
Sample	sample	`subject` and `email`
Sample Delta	sampleDelta	`subject` and `email`

- Three entries on **Applications** > **Integration** > **Target URL Mapping**.

URL	Target Session
https://sso.xray.local:9031/SpSample/MainPage?app=Alpha&*	Sample
https://sso.xray.local:9031/SpSample/MainPage?app=Charlie&*	Sample
https://sso.xray.local:9031/SpSample/MainPage?app=Delta&*	Sample Delta

- Three IdP connections to your partners.

Partner(Federation ID)	Identity Mapping	Attribute Contract	Target Session MappingSP adapter instance name(SP adapter instance ID)
Alpha (sso.alpha.local)	Account Mapping	`SAML_SUBJECT` and `samlEmail`	Sample (sample)

Partner(Federation ID)	Identity Mapping	Attribute Contract	Target Session MappingSP adapter instance name(SP adapter instance ID)
Charlie (sso.charlie.local)	Account Mapping	`SAML_SUBJECT` and `samlEmail`	Sample (sample)
Delta (sso.delta.local)	Account Mapping	`SAML_SUBJECT` and `samlEmail`	Sample Delta (sampleDelta)

In this example, all partners support SAML 2.0 and only the SP-initiated single sign-on (SSO) profile.

- SP-initiated SSO URLs for users from Alpha, Charlie, and Delta.

Partner	SSO URL
Alpha	https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.alpha.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%
Charlie	https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.charlie.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage
Delta	https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.delta.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%

After upgrading to PingFederate, you have the following new requirements:

- Create new IdP connections to three new partners: Echo, Foxtrot and Golf.
- Enforce multi-factor authentication (MFA) for users from Alpha, Charlie, Echo, and Golf through Bravo.

  Bravo requires a user ID to be passed in from the original source and returns only the user ID when the users fulfill the multi-factor authentication (MFA) requirement.

The new required components are:

- Two additional SP adapter instances. For more information, see step 1:
  - **Sample Echo** to integrate with Echo's target application.
  - **Sample Golf** to integrate with Golf's target application.
- Four new IdP connections. For more information, see step 2, step 3, and step 4:

Partner(Federation ID)	Identity Mapping	Attribute Contract	Target Session MappingSP adapter instance name(SP adapter instance ID)
Bravo (sso.bravo.local)	No Mapping	`SAML_SUBJECT` and no other attributes	N/A
Echo (sso.echo.local)	No Mapping	`SAML_SUBJECT` and `samlEmail`	N/A
Foxtrot (sso.foxtrot.local)	Account Mapping	`SAML_SUBJECT` and `samlEmail`	Sample (sample)
Golf (sso.golf.local)	No Mapping	`SAML_SUBJECT` and `samlEmail`	N/A

In this example, all partners support SAML 2.0 and only the SP-initiated SSO profile.

- Three authentication policy contracts. For more information, see step 5:
  - An authentication policy contract, **Authenticated**, to carry user attributes from Alpha and Charlie to their respective target applications.
  - Two other authentication policy contracts, **Echo authenticated** and **Golf authenticated**, to carry user attributes from Echo and Golf to their target applications.
- An instance of the HTTP Request Parameter Authentication Selector, **PartnerIdpId**, to determine if a request is meant for Alpha or Charlie, because Alpha's and Charlie's target applications share an SP adapter instance. For more information, see step 6.
- Three SP authentication policies to enforce the multifactor authentication requirement. For more information, see step 7, step 8, and step 12.
- Three adapter mappings between the authentication policy contracts and the applicable SP adapter instances. For more information, see step 9:
  - Map from **Authenticated** to **Sample**.
  - Map from **Echo authenticated** to **Sample Echo**.
  - Map from **Golf authenticated** to **Sample Golf**
- Three additional target URL mappings between the applications requested by users from Echo, Foxtrot, and Golf to their respective SP adapter instances. For more information, see step 10:
- SSO URLs for all partners. For more information, see step 11.

Follow these steps to fulfill the new requirements:

Steps

1. Go to **Applications** > **Integration** > **SP Adapters**. On the **SP Adapters** window, create two new SP adapter instances, as shown in the following table.

Instance Name	Instance ID	Extended Contract
Sample Echo	sampleEcho	`subject` and `email`
Sample Golf	sampleGolf	`subject` and `email`

2. Create an IdP connection to Bravo.

    1. On the **IdP Connections** window (**Authentication** > **Integration** > **IdP Connections**) create a SAML 2.0 IdP connection.
    2. On **IdP Connection** > **Browser SSO** > **SAML Profiles**, select the **SP-Initiated SSO** check box.
    3. On **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Identity Mapping**, select **No Mapping**.
    4. Complete the rest of the connection configuration.

3. Create IdP connections to Echo and Golf.

    1. Go to **Authentication** > **Integration** > **IdP Connections**. On the**IdP Connections** window, create a SAML 2.0 IdP connection to Echo, and then to Golf.
    2. On **IdP Connection** > **Browser SSO** > **SAML Profiles**, select the **SP-Initiated SSO** check box.
    3. On **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Identity Mapping**, select **No Mapping**.

    > **Tip**
    >
    > If the partner and your organization agree to support account linking, select **Account Linking**. When the **Account Linking** option is selected, you must map the applicable SP adapter instance, **Sample Echo** for Echo or **Sample Golf** for Golf, to the connection on **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Target Session Mapping**. The rest of the steps remain unchanged.
    > This example does not use account linking.

    4. On **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Attribute Contract**, enter `samlEmail` under **Extend the Contract**.
    5. Complete the rest of the connection configuration.
    6. Repeat these steps to create a SAML 2.0 IdP connection to Golf.

4. Create an IdP connection to Foxtrot.

    1. On the **IdP Connections** window, create a SAML 2.0 IdP connection.
    2. On **IdP Connection** > **Browser SSO** > **SAML Profiles**, select the **SP-Initiated SSO** check box.
    3. On the **User-Session Creation** tab, click **Configure User-Session Creation**.
    4. On the **Identity Mapping** tab, select **Account Mapping**.

5. On the **Attribute Contract** tab, enter `samlEmail` under **Extend the Contract**.

6. On the **Target Session Mapping** tab, click **Map New Adapter Instance** and follow the **Adapter Mapping & User Lookup** configuration workflow to map the attributes from the assertion to the SP adapter instance **Sample**.

   Example:

Adapter Contract	Source	Value
email	Assertion	`samlEmail`
subject	Assertion	`SAML_SUBJECT`

7. Complete the rest of the connection configuration.

5. Create three authentication policy contracts, one for Alpha and Charlie, one for Echo, and one for Golf.

> **Note**
>
> The purpose of an authentication policy contract is to harness user attributes obtained through one or more authentication sources as the request flows through the applicable authentication policy. It is the medium between the authentication policies and the target applications. In general:
>
> - You map attributes to authentication policy contracts from authentication policies. For more information, seestep 7 in this example.
> - You map attributes from authentication policy contracts to target applications through adapter mappings. For more information, see step 9 in this example.

1. Go to **Authentication** > **Policies** > **Policy Contracts**. On the **Policy Contracts** window, click **Create New Contract** to create an authentication contract for users from Alpha and Charlie, for users from Echo, and then for users from Golf.

2. On the **Contract Info** tab, enter a name for this authentication policy contract.

   In this example, the names are `Authenticated`, `Echo authenticated`, and `Golf authenticated`.

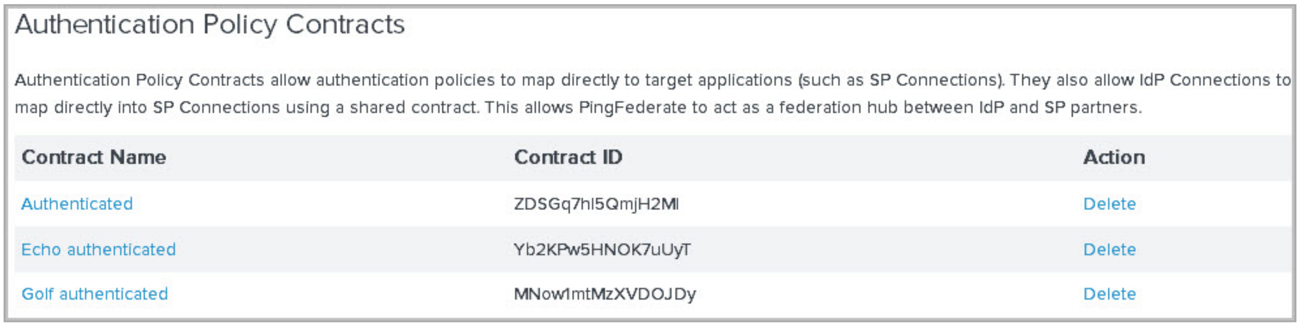
3. On the **Contract Attributes** tab, extend the authentication policy contract with an attribute for user's email address, such as `mail`.

4. Complete the rest of the connection configuration.

5. Repeat these steps to create an authentication policy contract for users from Echo, and then for users from Golf.

   Result:

   Your policy contracts should be similar to the following samples.



Authentication Policy Contracts

Authentication Policy Contracts allow authentication policies to map directly to target applications (such as SP Connections). They also allow IdP Connections to map directly into SP Connections using a shared contract. This allows PingFederate to act as a federation hub between IdP and SP partners.

Contract Name	Contract ID	Action
Authenticated	ZDSGq7hl5QmjH2Ml	Delete
Echo authenticated	Yb2KPw5HNOK7uUyT	Delete
Golf authenticated	MNow1mtMzXVDOJDy	Delete

6.

> **Note**
>
> If multiple target applications share the same SP adapter instance, you must use a selector to evaluate the SP-initiated requests such that the policy engine can route the requests to the policy paths that are meant for the respective IdPs. This example uses an instance of the HTTP Request Parameter Authentication Selector to categorize requests based on their respective `PartnerIdpId` query parameter values at runtime. The policy diverts accordingly based on the selector results.

Create an instance of the HTTP Request Parameter Authentication Selector.

1. Go to **Authentication** > **Policies** > **Selectors**. On the **Selectors** window, click **Create New Instance**.
2. On the **Type** tab, select **HTTP Request Parameter Authentication Selector** from the **Type** list and provide a name and ID for the selector instance.

In this example, the name and ID are both `PartnerIdpId`.

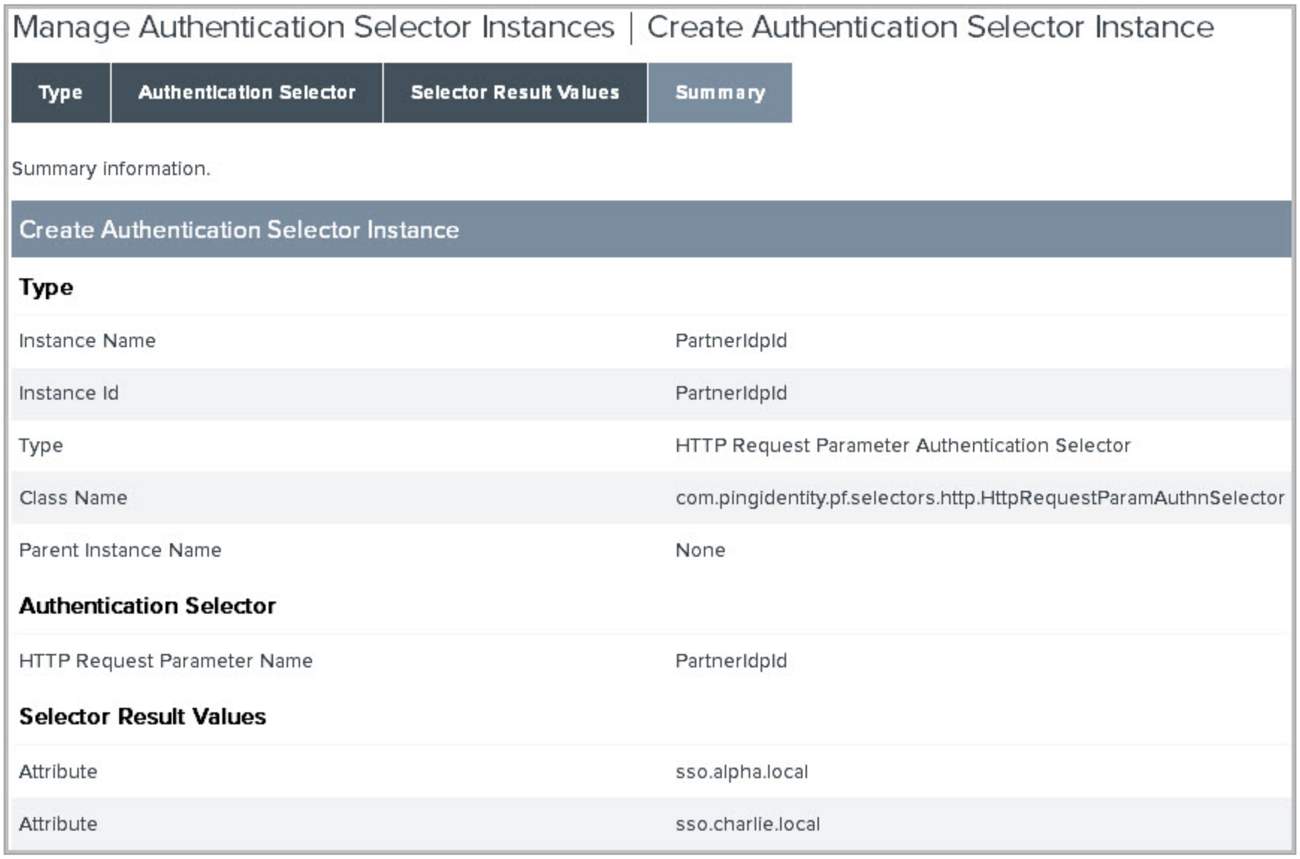
1. On the **Authentication Selector** tab, enter `PartnerIdpId` in the **HTTP Request Parameter Name** field.
2. On the **Selector Result Values** tab, enter `sso.alpha.local` and `sso.charlie.local` as the result values.

   > **Note**
   >
   > In general, for the IdPs that you want to enforce additional authentication requirements through one or more SP authentication policies and whose target applications share an SP adapter instance, you must enter their federation IDs here.

3. Complete the rest of the configuration.

   Result:

   Your selector instance should be similar to the following sample.



Manage Authentication Selector Instances | Create Authentication Selector Instance

Summary information.

Create Authentication Selector Instance

Type

Instance Name	PartnerIdpId
Instance Id	PartnerIdpId
Type	HTTP Request Parameter Authentication Selector
Class Name	com.pingidentity.pf.selectors.http.HttpRequestParamAuthnSelector
Parent Instance Name	None

Authentication Selector

HTTP Request Parameter Name	PartnerIdpId

Selector Result Values

Attribute	sso.alpha.local
Attribute	sso.charlie.local

7. Go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, click **Add Policy** to define a policy to enforce the third-party authentication requirement for users from Echo, and then for users from Golf.

> **Tip**
>
> If you need more information about each sub step, see step 4 in Configuring an SP authentication policy for users from one IdP.

1. On the **Policy** window, enter a name and a description for this policy, and select **sso.echo.local (IdP Connection)** as the first policy action from the list.

2. For the **sso.echo.local** > **Fail** path, select **Done** as the policy action.

3. For the **sso.echo.local** > **Success** path, select **sso.bravo.local (IdP Connection)** as the policy action.

4. Click **Options**, underneath **sso.bravo.local (IdP Connection)**, to relay the user ID (`SAML_SUBJECT`) from `sso.echo.local` to `sso.bravo.local` .

5. For the **sso.bravo.local** > **Fail** path, select **Done** as the policy action.

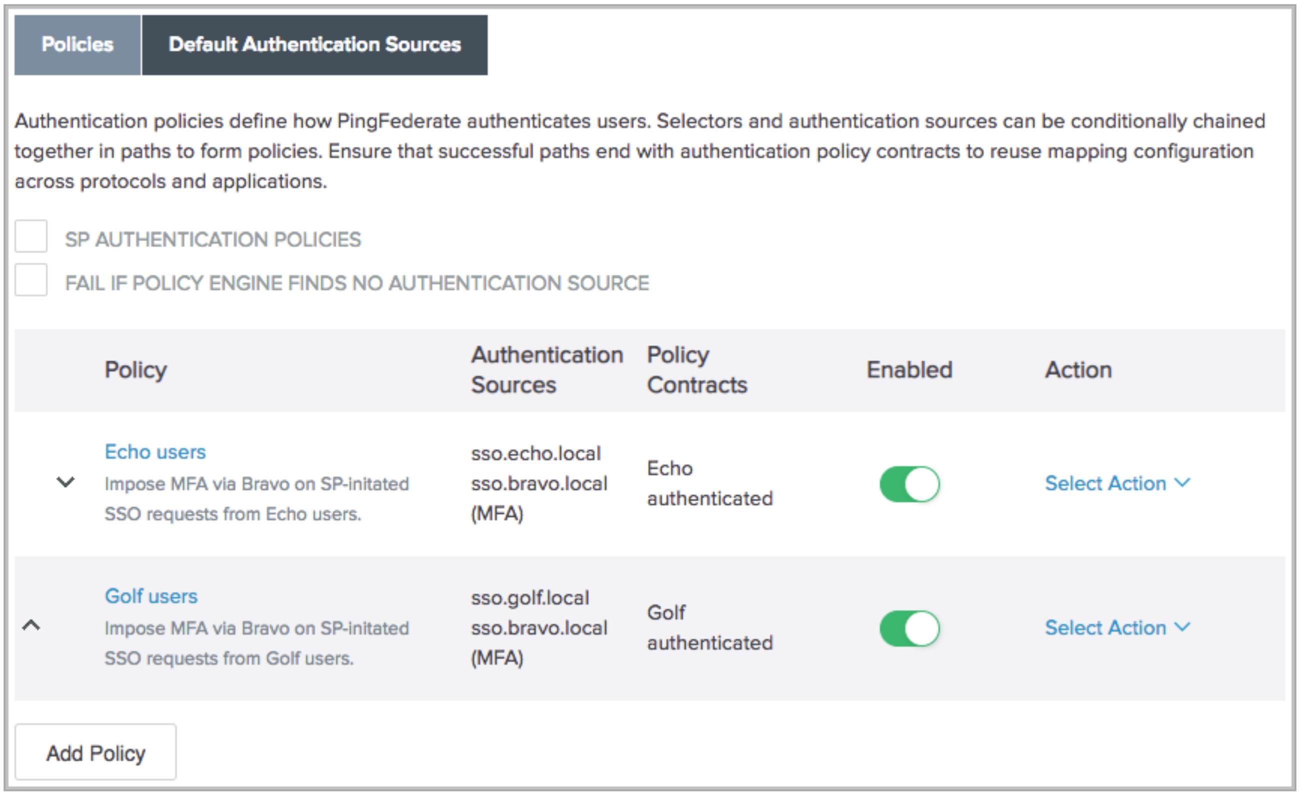
6. For the **sso.bravo.local** > **Success** path, select the policy contract **Echo authenticated** as the policy action, and then click **Contract Mapping**, underneath the policy contract, to configure the fulfillment of the policy contract.

> **Tip**
>
> Essentially, you are mapping attributes to an authentication policy contract from the authentication policy.

7. Repeat these steps to define a new authentication policy to enforce third-party authentication for users from Golf.

   Result:

   Your policies should be similar to the following samples.



Policies | Default Authentication Sources

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

□ SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Echo users Impose MFA via Bravo on SP-initated SSO requests from Echo users.	sso.echo.local sso.bravo.local (MFA)	Echo authenticated	●	Select Action
Golf users Impose MFA via Bravo on SP-initated SSO requests from Golf users.	sso.golf.local sso.bravo.local (MFA)	Golf authenticated	●	Select Action

Add Policy

8. On the **Policies** tab, click **Add Policy** to define a new authentication policy to enforce third-party authentication for users from Alpha and Charlie.

> **Note**
>
> If multiple target applications share the same SP adapter instance, you must use a selector to evaluate the SP-initiated Browser SSO requests, such that the policy engine can route the requests to the policy paths that are meant for the respective IdPs. This example uses an instance of the HTTP Request Parameter Authentication Selector created in step 6.

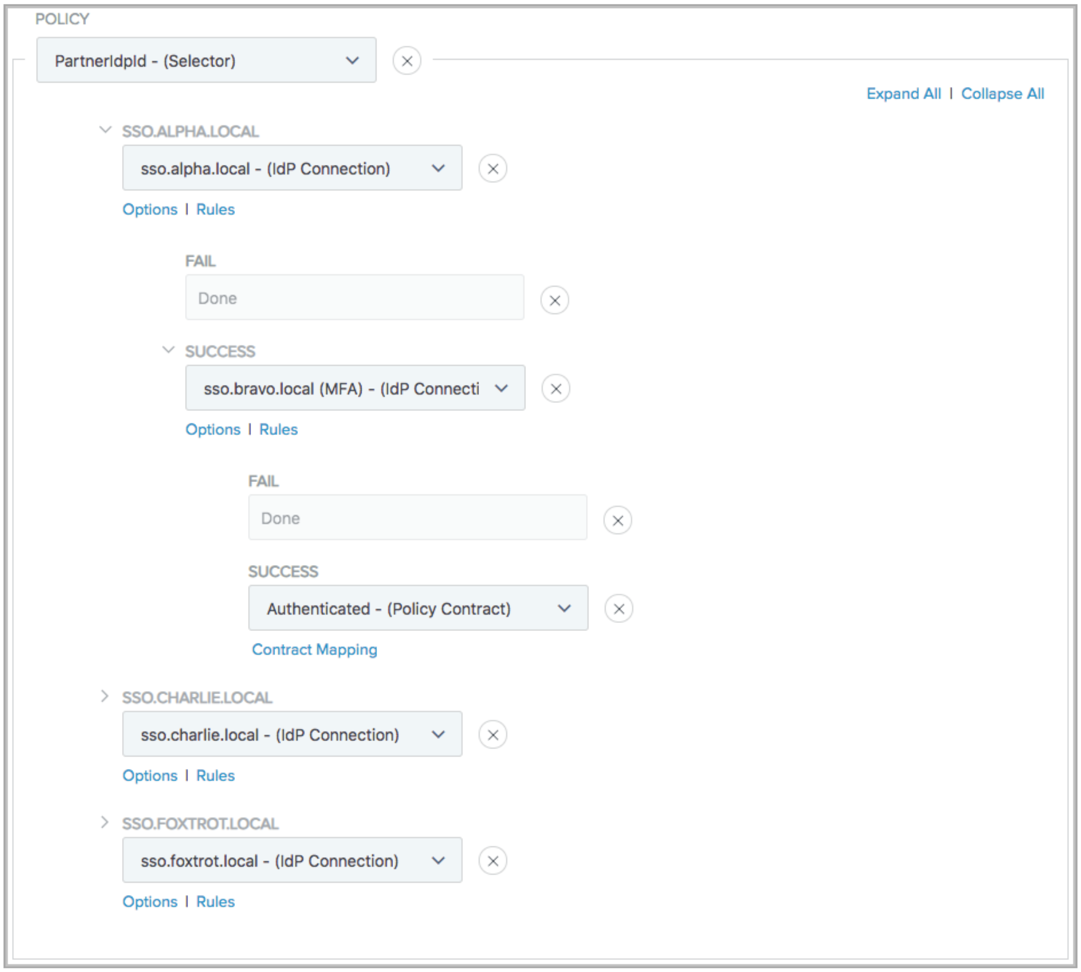
   1. On the **Policy** window, enter a name and a description for this policy, and select the instance of the HTTP Request Parameter Authentication Selector as the first policy action from the list.
   2. For the **sso.alpha.local** path, repeat step 7 to configure the authentication requirement for the `sso.alpha.local` IdP connection.

3. For the **sso.charlie.local** path, repeat step 7 to configure the authentication requirement for the `sso.charlie.local` IdP connection.

4. For the **sso.foxtrot.local** path, repeat step 7 to configure the authentication requirement for the `sso.foxtrot.local` IdP connection.

   Result:

   Your policy should be similar to the following sample.



POLICY

PartnerIdpId - (Selector)

Expand All | Collapse All

SSO.ALPHA.LOCAL

sso.alpha.local - (IdP Connection)

Options | Rules

FAIL

Done

SUCCESS

sso.bravo.local (MFA) - (IdP Connecti

Options | Rules

FAIL

Done

SUCCESS

Authenticated - (Policy Contract)

Contract Mapping

SSO.CHARLIE.LOCAL

sso.charlie.local - (IdP Connection)

Options | Rules

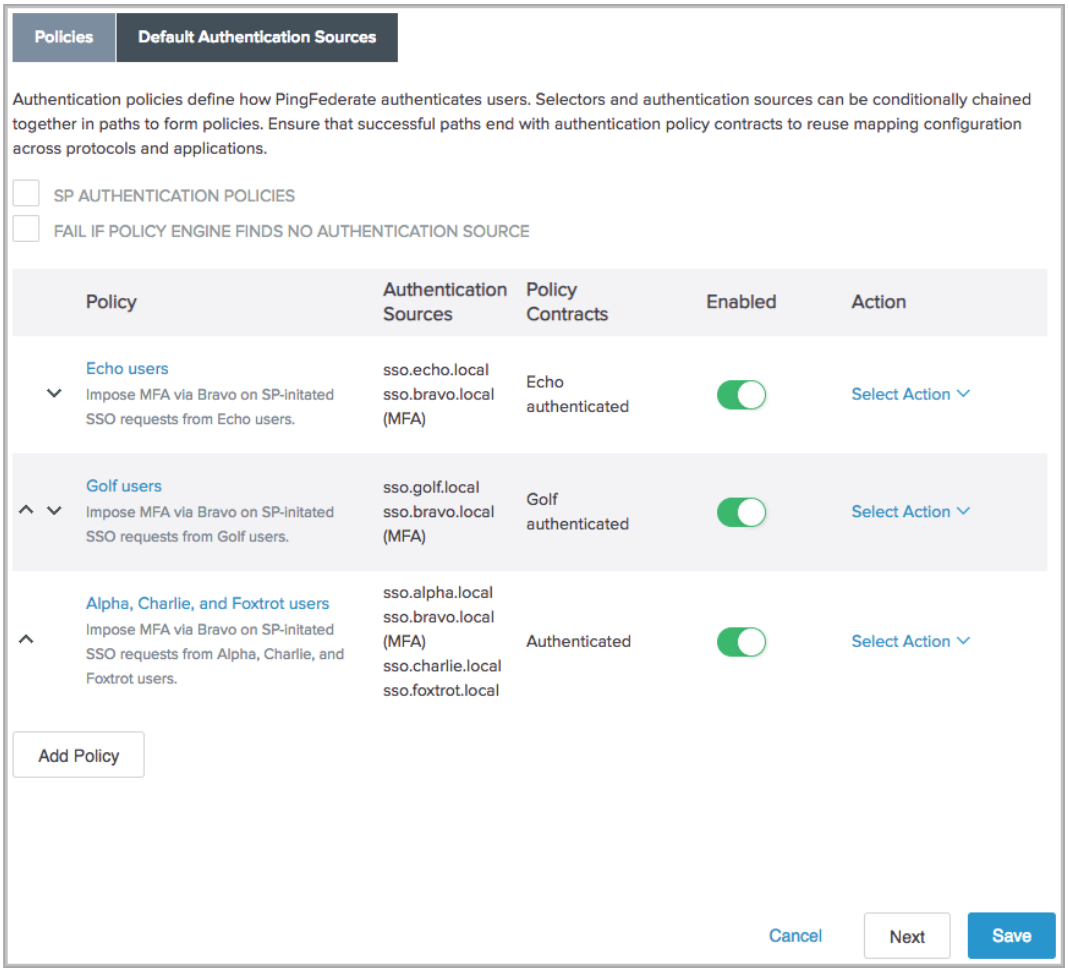
SSO.FOXTROT.LOCAL

sso.foxtrot.local - (IdP Connection)

Options | Rules

> **Note**
>
> This window capture collapses the **sso.charlie.local** and **sso.foxtrot.local** policy paths for documentation presentation purposes.

When you go back to the **Policies** tab, your policies should be similar to the following samples.



Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

□ SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Echo users Impose MFA via Bravo on SP-initated SSO requests from Echo users.	sso.echo.local sso.bravo.local (MFA)	Echo authenticated	●	Select Action
Golf users Impose MFA via Bravo on SP-initated SSO requests from Golf users.	sso.golf.local sso.bravo.local (MFA)	Golf authenticated	●	Select Action
Alpha, Charlie, and Foxtrot users Impose MFA via Bravo on SP-initated SSO requests from Alpha, Charlie, and Foxtrot users.	sso.alpha.local sso.bravo.local (MFA) sso.charlie.local sso.foxtrot.local	Authenticated	●	Select Action

5. Click **Save**.

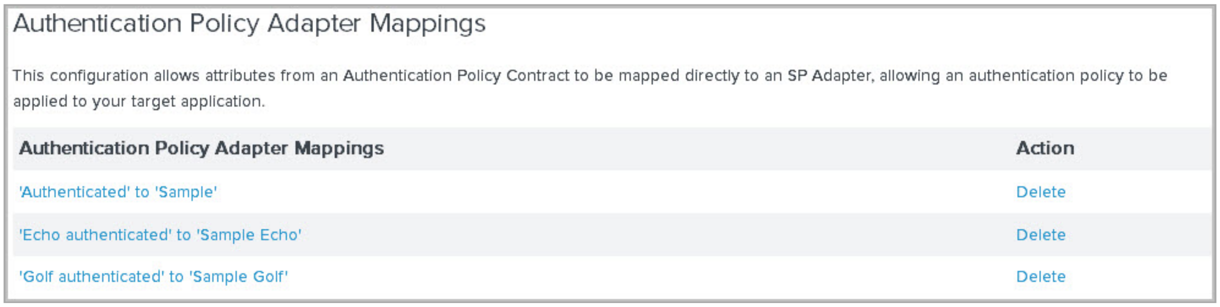
9. Create adapter mappings.

1. Go to **Applications** > **Integration** > **Policy Contract Adapter Mappings**. On the **Policy Contracts** window, create three adapter mappings to map from the authentication policy contracts, created in step 5, to the applicable SP adapter instances.

   - Map from **Authenticated** to **Sample**.
   - Map from **Echo authenticated** to **Sample Echo**.
   - Map from **Golf authenticated** to **Sample Golf**.

> **Tip**
>
> If you need more information about each sub step, see step 5 in Configuring an SP authentication policy for users from one IdP.

Your mappings should be similar to the following samples.

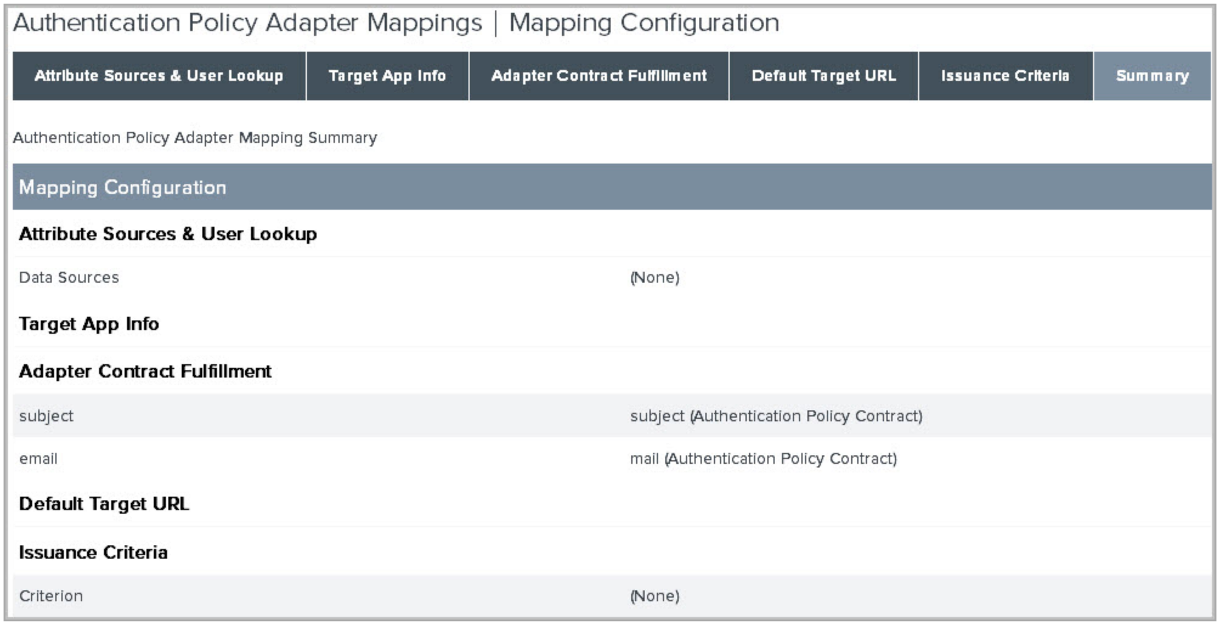


Authentication Policy Adapter Mappings

This configuration allows attributes from an Authentication Policy Contract to be mapped directly to an SP Adapter, allowing an authentication policy to be applied to your target application.

Authentication Policy Adapter Mappings	Action
'Authenticated' to 'Sample'	Delete
'Echo authenticated' to 'Sample Echo'	Delete
'Golf authenticated' to 'Sample Golf'	Delete

Each adapter mapping should be similar to the following configuration.



Authentication Policy Adapter Mappings | Mapping Configuration

Authentication Policy Adapter Mapping Summary

Mapping Configuration

Attribute Sources & User Lookup

Data Sources	(None)

Target App Info

Adapter Contract Fulfillment

subject	subject (Authentication Policy Contract)
email	mail (Authentication Policy Contract)

Default Target URL

Issuance Criteria

Criterion	(None)

Tip

You are essentially mapping attribute values from the authentication policy contracts to target applications through the applicable SP adapter instances.

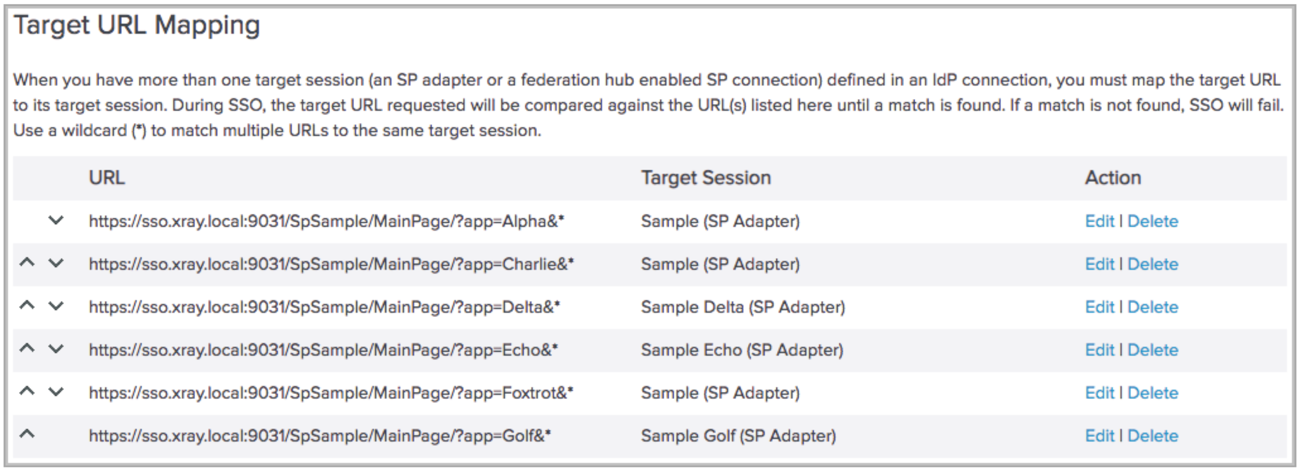
10. Create target URL mappings.

    1. Go to **Applications** > **Integration** > **Target URL Mapping**. On the **Target URL Mapping** window, add the following mappings.

URL	Target Session
https://sso.xray.local:9031/SpSample/MainPage/?app=Echo&*	**Sample Echo**
https://sso.xray.local:9031/SpSample/MainPage/?app=Foxtrot&*	**Sample**
https://sso.xray.local:9031/SpSample/MainPage/?app=Golf&*	**Sample Golf**

Result:

Your target URL mappings should be similar to the following samples.



Target URL Mapping

When you have more than one target session (an SP adapter or a federation hub enabled SP connection) defined in an IdP connection, you must map the target URL to its target session. During SSO, the target URL requested will be compared against the URL(s) listed here until a match is found. If a match is not found, SSO will fail. Use a wildcard (*) to match multiple URLs to the same target session.

URL	Target Session	Action
https://sso.xray.local:9031/SpSample/MainPage/?app=Alpha&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Charlie&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Delta&*	Sample Delta (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Echo&*	Sample Echo (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Foxtrot&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Golf&*	Sample Golf (SP Adapter)	Edit | Delete

11. Configure the following SSO URLs.

Partner	SSO URL
Alpha	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.alpha.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DAlpha%26t%3Da` The SSO URL has not changed. Based on the current configuration, because the target applications for Alpha and Charlie share the same SP adapter instance, the `PartnerIdpId` query parameter is required for the configured policy to route the request to the corresponding IdP connection.
Charlie	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.charlie.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DCharlie%26t%3Dc` The SSO URL has not changed. Based on the current configuration, because the target applications for Alpha and Charlie share the same SP adapter instance, the `PartnerIdpId` query parameter is required for the configured policy to route the request to the corresponding IdP connection.
Delta	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.delta.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DDelta%26t%3Dd` The SSO URL has not changed. Optionally, based on the current configuration, you can remove the `PartnerIdpId` query parameter because it is not required. You can also replace the `TargetResource` query parameter and its value with the `SpSessionAuthnAdapterId` query parameter and the applicable SP adapter instance ID, `SpSessionAuthnAdapterId=sampleDelta`, if you have configured a default target URL in the IdP connection to Delta, or a default SP SSO URL for all IdP connections.

Partner	SSO URL
Echo	`https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleEcho&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DEcho%26t%3De` This is a new SSO URL. Optionally, based on the current configuration, you can remove the `TargetResource` query parameter and its value if you have configured a default target URL in the IdP connection to Echo, or a default SP SSO URL for all IdP connections.
Foxtrot	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.foxtrot.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DFoxtrot%26t%3Df` This is a new SSO URL. Optionally, based on the current configuration, you can remove the `TargetResource` query parameter and its value if you have configured a default target URL in the IdP connection to Foxtrot, or a default SP SSO URL for all IdP connections.
Golf	`https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleGolf&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DGolf%26t%3Dg` This is a new SSO URL. Optionally, based on the current configuration, you can remove the `TargetResource` query parameter and its value if you have configured a default target URL in the IdP connection to Golf, or a default SP SSO URL for all IdP connections.

> **Important**
>
> When using the parameters `TargetResource` or `TARGET` with their own query parameters included, the parameter value must be URL-encoded. Any other parameters that contain restricted characters, such as many SAML URNs, also must be URL-encoded. For information about URL encoding, see third party resources such as HTML URL-encoding Reference. Parameters are case-sensitive.

12. To test your new use case, go to **Authentication** > **Policies** > **Policies**. On the **Policies** window, select the **SP Authentication Policies** check box to enable policies for SP-initiated Browser SSO requests received by at the `/sp/startSSO.ping` endpoint, and click **Save**.

Configuring SP authentication policies for internal users

The `/pf/adapter2adapter.ping` endpoint initiates direct IdP-to-SP adapter mapping, mostly intended for the internal users to access resources without maintaining a service provider (SP) and an identity provider (IdP) connection on the same server.

About this task

To prevent users from circumventing the SP authentication policies, this endpoint becomes inactive when SP authentication policies are enabled but IdP authentication policies are disabled. Administrators can configure SP authentication policies for the internal users to re-enable access to protected resources.

Suppose you have configured the following use cases in PingFederate 8.0.

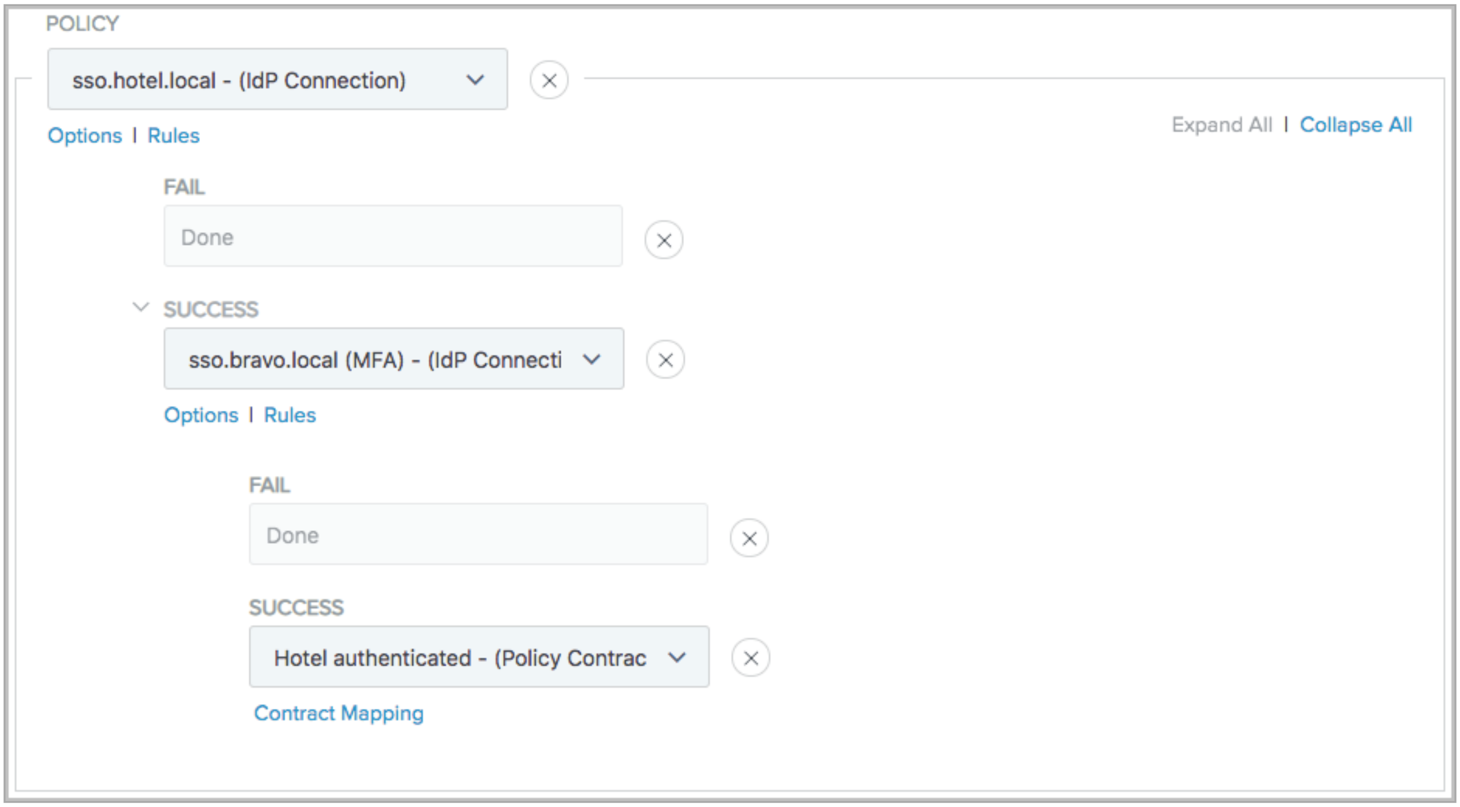
For users from Hotel, an IdP:

- An SP adapter instance:
  - Name: Sample Hotel
  - ID: sampleHotel
- A SAML 2.0 IdP connection:
  - Partner: Hotel
  - Federation ID: sso.hotel.local
  - SAML Profile: SP-initiated single sign-on (SSO) only
  - Identity mapping method: Account mapping
  - Default target URL: https://sso.xray.local:9031/SpSample/MainPage/?app=Hotel&t=h
  - SSO URL: https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.hotel.local

For internal users:

- An IdP HTML Form Adapter instance, HTML Form, validating credentials through a Password Credential Validator (PCV) instance against your user directory
- An adapter-to-adapter mapping:
  - Source: HTML Form
  - Target: Sample Hotel
  - Default target URL: https://sso.xray.local:9031/SpSample/MainPage/?app=Internal&t=i
  - SSO URL: https://sso.xray.local:9031/pf/adapter2adapter.ping?SpSessionAuthnAdapterId=sampleHotel

After upgrading your version of PingFederate, if you want to enforce multi-factor authentication fousers from Hotel through Bravo, you can create an IdP connection to Bravo and the following authentication policy.


POLICY
sso.hotel.local - (IdP Connection)
Options | Rules
Expand All | Collapse All
FAIL
Done
SUCCESS
sso.bravo.local (MFA) - (IdP Connecti
Options | Rules
FAIL
Done
SUCCESS
Hotel authenticated - (Policy Contrac
Contract Mapping

Because the authentication policy ends with a policy contract **Hotel authenticated**, you must create an adapter mapping, from the policy contract, to **Sample Hotel**, the SP adapter instance integrated with the target application. You also need to update the SSO URL for users from Hotel to https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleHotel.

When you select the **SP Authentication Policies** check box without selecting the **IdP Authentication Policies** check box, the `/pf/adapter2adapter.ping` endpoint is disabled to prevent malicious Hotel's users, with specific knowledge of PingFederate endpoints, your PingFederate configuration, and functional credentials, from trying to access the target application through the SSO URL intended for your internal users. By doing so, you circumvent the SP authentication policy that is meant for them.

To re-enable access to the application for the internal users, you need the following new components:

- An additional SP adapter instance, **Sample Internal** to integrate with the target application. See step 1.
- An authentication policy contract, **Internal authenticated**, to carry attributes from internal users to the target applications. See step 3.
- An instance of the CIDR Authentication Selector to be deployed in the authentication policy to reject users from external networks to access protected resources using your HTML Form Adapter. Alternatively, deploy an instance of the PingID Adapter in the authentication policy to enforce multifactor authentication for users who authenticated successfully using the HTML Form Adapter. See step 5.
- An authentication policy for the internal users. See step 7.
- An adapter mapping to map from the authentication policy contracts **Internal authenticated** to the SP adapter instance **Sample Internal**. See step 8.

- A new SSO URL for the internal users. See step 11.

Follow these steps to fulfill the new requirements.

Steps

1. Go to **Applications** > **Integration** > **SP Adapters**.

2. Click **Create New Instance** to create a new SP adapter instance.

   For each new instance, include the name, ID, and extended contract.

Instance Name	Instance ID	Extended Contract
Sample Internal	sampleInternal	`subject` and `email`

3. Go to **Authentication** > **Policies** > **Policy Contracts**.

4. Click **Create New Contract** to create an authentication policy contract.

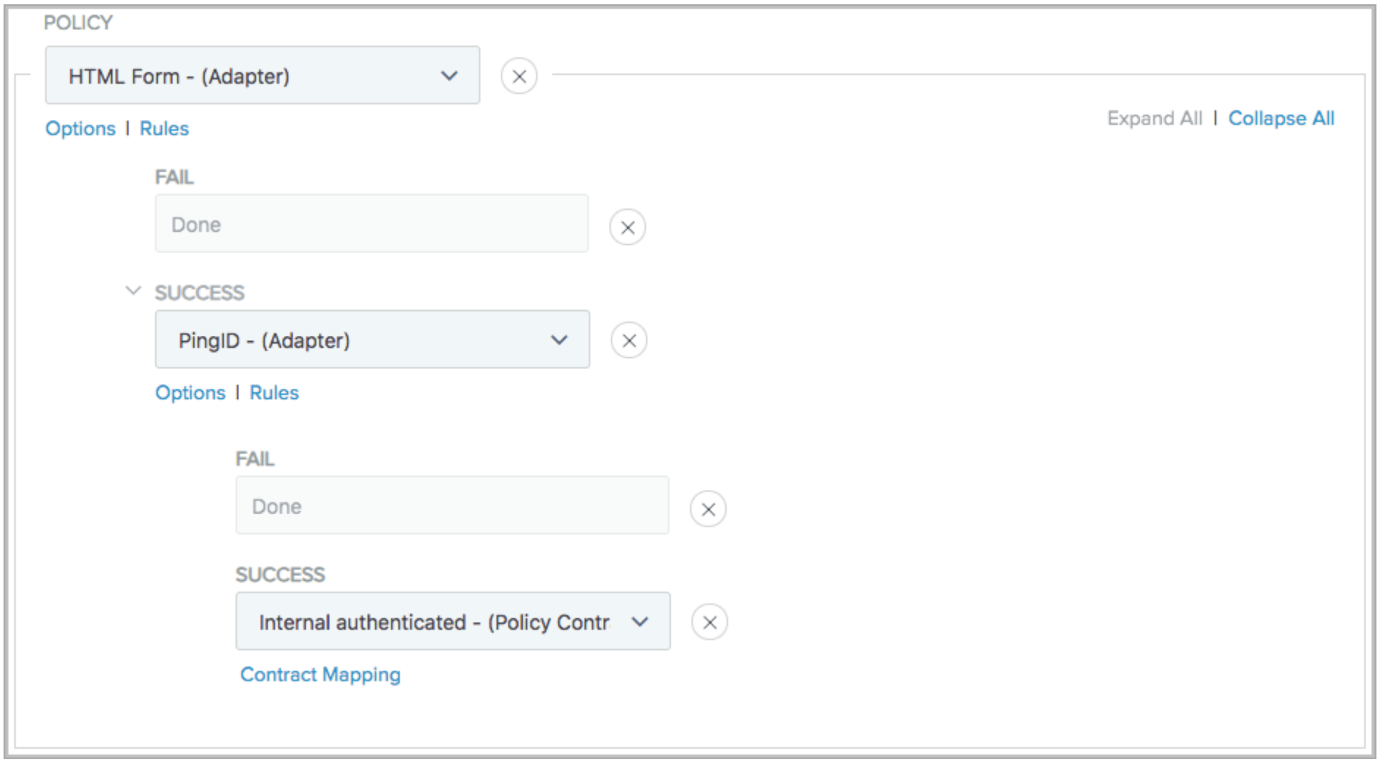
   > **Note**
   >
   > In this example, the name of the policy contract is `Internal authenticated.`
   > The purpose of an authentication policy contract is to harness user attributes obtained through one or more authentication sources as the request flows through the applicable authentication policy. It is the medium between the authentication policies and the target applications. Generally speaking:
   >
   > - You map attributes to authentication policy contracts from authentication policies. step 7 in this example.
   > - You map attributes from authentication policy contracts to target applications through adapter mappings, step 8 in this example.

5. Go to **Authentication** > **Policies** > **Selectors**.

6. Click **Create New Instance** to create an instance of the CIDR Authentication Selector with one or more network ranges that correspond to your internal users.

   > **Note**
   >
   > In this example, the name and ID are both `Internal users` .
   > The purpose of the CIDR Authentication Selector is for the policy engine to reject users from external networks to access protected resources using your HTML Form Adapter. You can also deploy the PingID Adapter to enforce multifactor authentication for users authenticated through the HTML Form Adapter. If users fail to fulfill the PingID multifactor authentication requirement, the policy engine rejects their requests, thus providing another layer of protection against unauthorized access from malicious users.

7. Go to **Authentication** > **Policies** > **Policies** to configure your policies as follows.



POLICY

HTML Form - (Adapter)

FAIL

Done

SUCCESS

PingID - (Adapter)

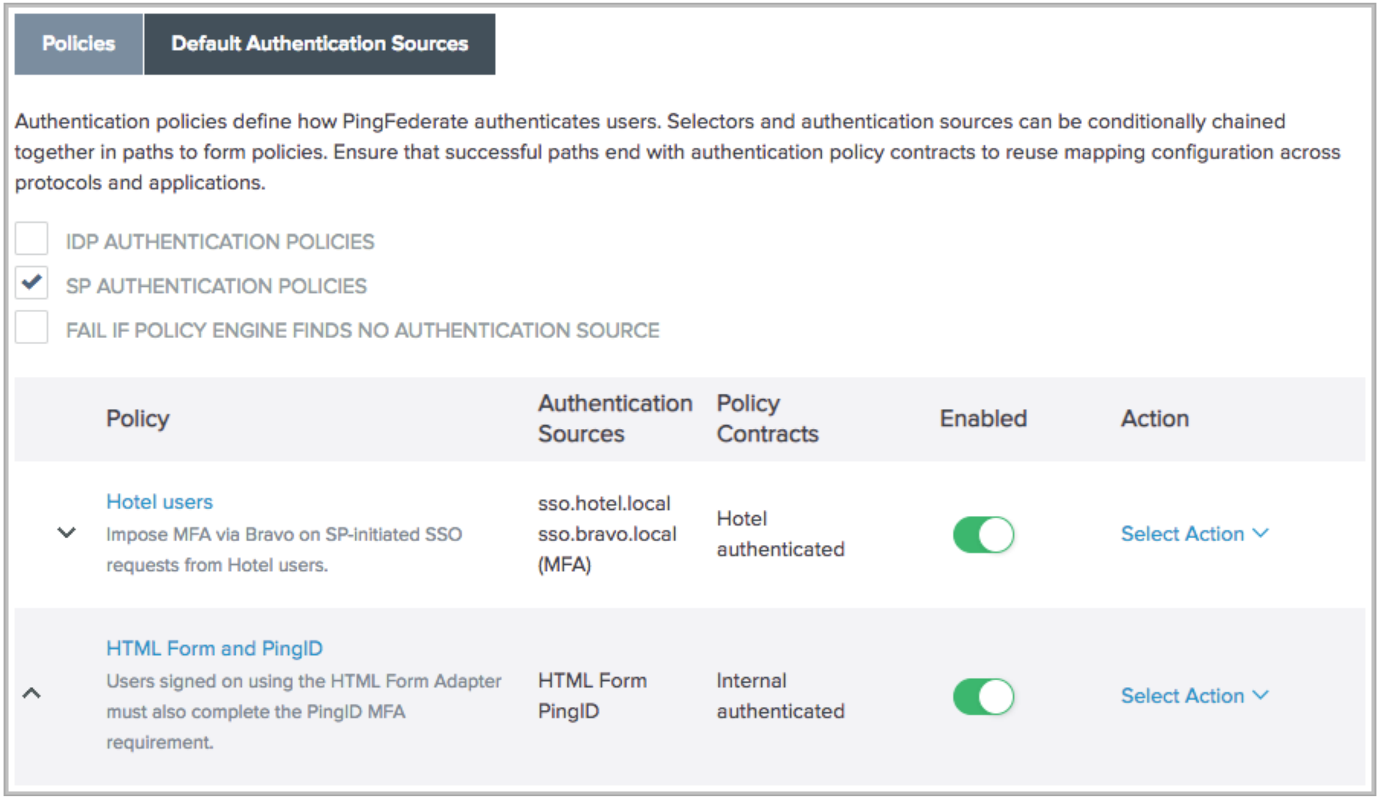
FAIL

Done

SUCCESS

Internal authenticated - (Policy Contr

With PingID Adapter



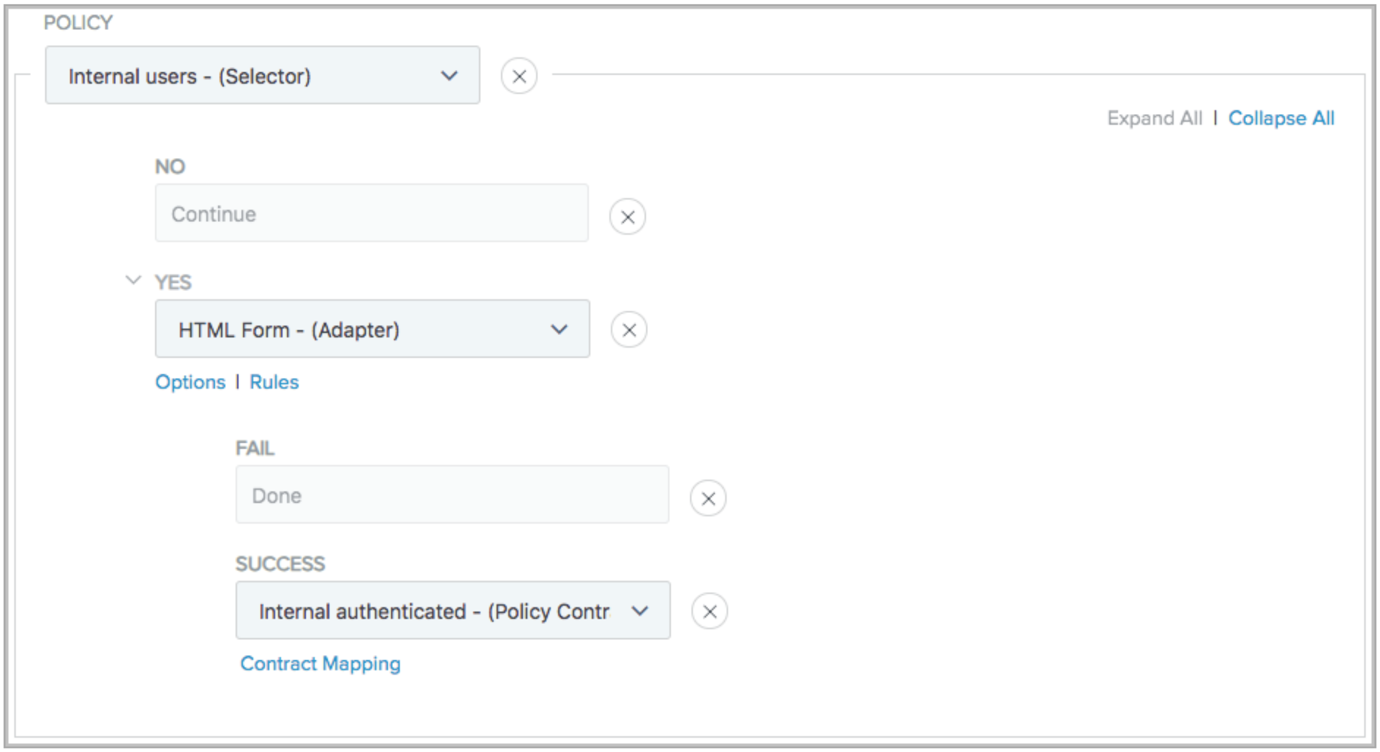
Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☐ IDP AUTHENTICATION POLICIES

☑ SP AUTHENTICATION POLICIES

☐ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Hotel users Impose MFA via Bravo on SP-initiated SSO requests from Hotel users.	sso.hotel.local sso.bravo.local (MFA)	Hotel authenticated	●	Select Action
HTML Form and PingID Users signed on using the HTML Form Adapter must also complete the PingID MFA requirement.	HTML Form PingID	Internal authenticated	●	Select Action



POLICY

Internal users - (Selector)

NO

Continue

YES

HTML Form - (Adapter)

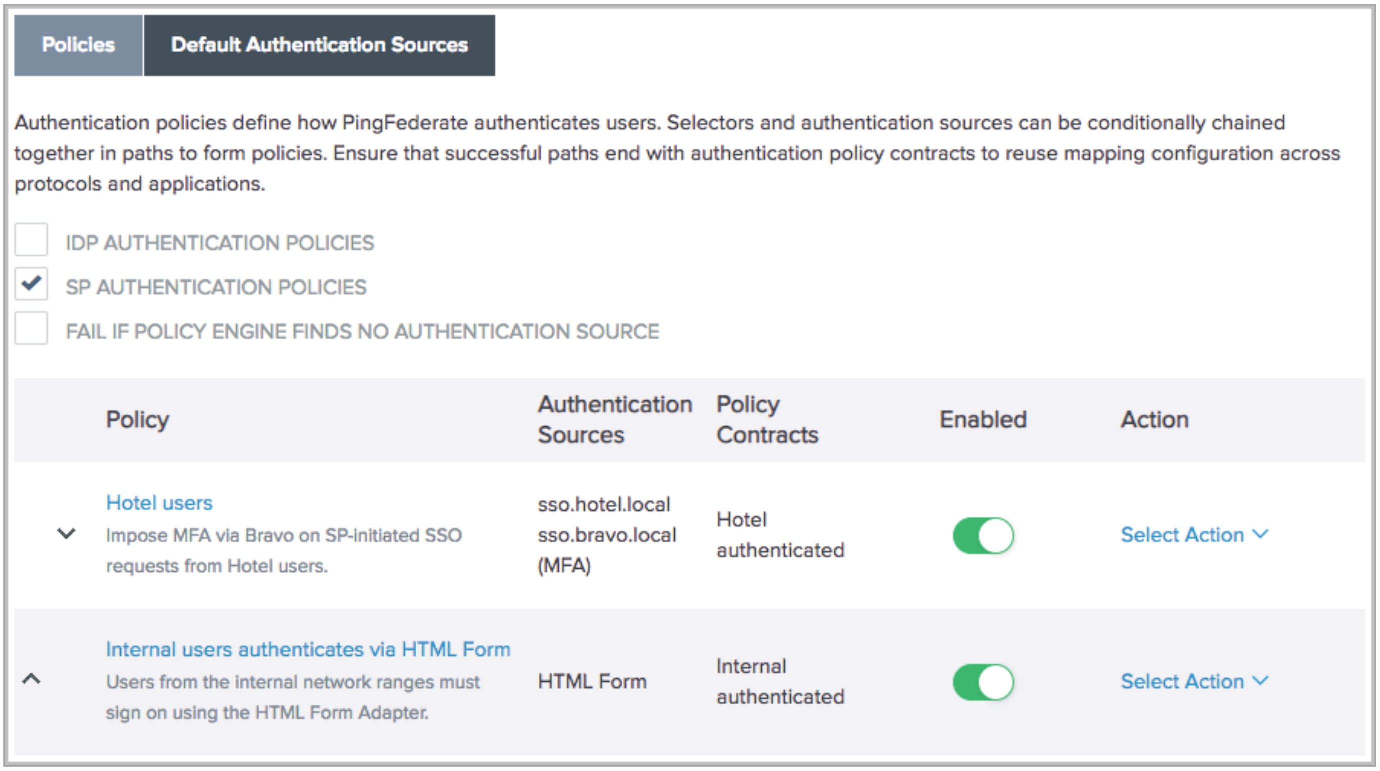
FAIL

Done

SUCCESS

Internal authenticated - (Policy Contr

With CIDR Authentication Selector



Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

□ IDP AUTHENTICATION POLICIES

☑ SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Hotel users Impose MFA via Bravo on SP-initiated SSO requests from Hotel users.	sso.hotel.local sso.bravo.local (MFA)	Hotel authenticated	●	Select Action
Internal users authenticates via HTML Form Users from the internal network ranges must sign on using the HTML Form Adapter.	HTML Form	Internal authenticated	●	Select Action

8. Go to **Applications** > **Integration** > **Policy Contract Adapter Mappings**.

9. Click **Create New Instance** and create a mapping from the new authentication policy contract, `Internal authenticated.` , to the new SP adapter instance, `sampleInternal` .

10. On the **Default Target URL** window, enter `https://sso.xray.local:9031/SpSample/MainPage/?app=Internal&t=i`.
11. Update the SSO URL for your internal users to https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleInternal.

Policy fragments

You can manage policy fragments under **Authentication** > **Policies** > **Fragments**. You can create reusable policy fragments and apply them in multiple authentication policies.

Fragments make policies easier to administer. They allow you to extract common patterns that exist among different policies and to manage them in one place. For example, you can create a reusable policy fragment with policy components that you frequently use and apply that fragment in multiple policies. When the authentication requirements need adjustments, you can make those changes in the fragment without updating the policies that reference it.

Fragments can use other fragments. You can chain up to five fragments together and use as many fragments as you want throughout a policy, but the policy cannot be configured to have over five fragments being processed at one time.

On the **Fragments** page:

- Click **Add Fragment** to define a policy fragment.
- Click the name of an existing fragment to edit its configuration.
- Click **Select Action** to copy or delete an existing fragment.

Defining policy fragments

You define policy fragments in the **Fragment** window.

Before you begin

Make sure that you have configured at least two policy contracts to function as input and output contracts.

Steps

1. In the **Name** field, type a name for the policy fragment.
2. **Optional:** Change the identifier for the fragment. This ID will be used to reference input and output attributes in the advanced Expressions fulfillment option. It cannot be changed after the fragment has been created.
3. **Optional:** Type a description for the fragment.
4. From the **Inputs** list, select the input authentication policy contract that calling members will need to fulfill. The attributes contained in the contract will be available for use throughout the policy.

5. From the **Outputs** list, select the output authentication policy contract that this fragment will fulfill. Calling members will be able to use the values of the attributes contained in the output policy contract.

6. From the **Policy** list, select an IdP adapter, an IdP connection, a selector, or a fragment. (Detailed policy configuration instructions are provided in step 5 in Defining authentication policies.)

   > **Important**
   >
   > You can select **Fragments** as the policy action and then select a policy fragment that you have created. When you select a fragment, click **Fragment Mapping** and use the in-product help links to access the topics that describe how to configure the mapping.

   1. Click **Options** and select the source and the attribute to be used as the incoming user ID.

   2. Click **Rules** and define authentication policy rules using attributes from the previous authentication source or from an earlier step in the policy. For more information, see Configuring rules in authentication policies.

   3. Configure **Fail** and **Success** paths. For a fragment to succeed, you must map it into a LIP or APC based on the output contract. You can also use a fragment in a calling policy and set both of the fragment’s exit Fail/Success nodes to Done.

      > **Note**
      >
      > The **Copy** and **Paste** feature lets you copy a policy path and paste it into another place in the same policy, another policy, or a policy fragment. After you copy and paste a path, follow the on-screen instructions to correct any errors.
      > One benefit of this feature is that you can easily add a new step at the start or middle of an existing policy. To do that, copy and then remove the existing path below the point where you will define the new step. After you define the new step, paste the copied path back into the policy below the new step.

7. Click **Save**.

Policy contracts

Authentication policy contracts, formerly known as connection mapping contracts, give PingFederate administrators a variety of capabilities.

Authentication policy contracts provide PingFederate administrators the following benefits:

- The capability to build an attribute contract with attribute values from multiple authentication sources or datastore queries through an authentication policy.
- The flexibility to map only the policy contract to a connection. Administrators do not have to map into the connection the authentication sources in the policy leading up to the contract. For example, administrators can experiment with various IdP adapter instances without the burden of adding and removing them to and from the connection.
- The potential to reuse authentication policies that use the same policy contract in multiple service provider (SP) connections, identity provider (IdP) connections, and OAuth use cases, using the OAuth Authorization Code or Implicit grant types.

Authentication policy contracts are also the media to carry user attributes from IdPs to SPs when PingFederate is deployed as a federation hub. For more information, see Federation hub use cases.

Managing policy contracts

You can manage authentication policy contracts in the **Authentication** > **Policies** > **Policy Contracts** window.

Steps

1. Click **Create New Contract** to create a new authentication policy contract.
2. Edit an existing authentication policy contract by clicking its name.
3. Check the policy contract usage.
4. **Optional:** On the **Policy Contracts** window, click **Delete** to remove an authentication policy contract, if the authentication policy contract is not in use, or cancel the removal request.

Editing contract information

You can create new policy contracts or edit existing ones from the **Policy Contracts** window.

Steps

1. Go to **Authentication** > **Policies** > **Policy Contracts**.
2. Click **Create New Contract** or select an existing contract.
3. On the **Contract Info** tab, enter or modify the contract name.
4. Click **Next**.

Defining contract attributes

To manage the user attributes in the authentication policy contract, go to **Authentication** > **Policies** > **Policy Contracts**.

About this task

Every authentication policy contract comes with a `subject` attribute. You can extend the contract with additional attributes as needed.

Steps

- If needed, enter an attribute under **Extend the Contract**, and then click **Add**.

> **Note**
>
> Attribute names are case-sensitive and must suit the needs of your partners. Repeat to add more attributes as needed.

- Click **Edit**, **Update**, or **Cancel** to modify an existing attribute or undo changes.
- Click **Delete** to remove an existing attribute.

Reviewing the policy contract

On the **Summary** tab, you can review the policy contract.

Steps

1. - To keep your changes, click **Save**.
   - To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
   - To discard your changes, click **Cancel**.

Special attribute names in contracts

PingFederate handles the `SAML_AUTHN_CTX` and `SAML_AUTHN_INSTANT` attribute names in specific ways.

SAML_AUTHN_CTX

The `SAML_AUTHN_CTX` attribute is involved in determining the authentication context value for the flow, which is mapped into the `acr` claim in the OpenID Connect (OIDC) ID token or the `AuthnContextClassRef` in the SAML assertion. The authentication context value represents the quality or type of authentication that was performed and may be used by downstream applications to decide what privileges to grant a user. The `AuthnContextClassRef` is required in SAML assertions and will be set to `urn:oasis:names:tc:SAML:1.0:am:unspecified` if a value can't be determined by the procedure below.

PingFederate determines the authentication context value by iterating over each authentication source that was invoked in the flow. Identity provider (IdP) adapters can return an authentication context value by including the `org.sourceid.saml20.adapter.idp.authn.authnCtx` attribute in the attribute map returned from `lookupAuthN()`.

For IdP connections, the authentication context value comes from the `acr` claim in the ID token or the `AuthnContextClassRef` value in the assertion. PingFederate also allows a mapping to be configured in the IdP connection between local and remote authentication context values under **Browser SSO > Protocol Settings > Overrides**.

PingFederate iterates over the authentication sources that were invoked, pulling an authentication context value from each one. The last authentication context value returned by an authentication source becomes the authentication context value for the overall flow.

Further control over the authentication context value for the flow is available by adding `SAML_AUTHN_CTX` to an authentication policy contract (APC). Any value mapped into this attribute overrides the authentication context value for the flow and will be passed to downstream applications through the ID token or assertion.

In some cases, you might want to populate the `SAML_AUTHN_CTX` in the APC with the value from a specific source. For SAML IdP connections, you can add `SAML_AUTHN_CTX` to the attribute contract. The corresponding attribute for OIDC IdP connections is `acr`. This will be automatically populated with the authentication context value coming from that IdP connection and you can then map from that attribute into `SAML_AUTHN_CTX` in the APC.

For IdP adapters that return an authentication context value, you can do the same thing by adding `org.sourceid.saml20.adapter.idp.authn.authnCtx` to the IdP adapter contract.

> **Note**
>
> Mapping into the `org.sourceid.saml20.adapter.idp.authn.authnCtx` attribute in the adapter's internal **Adapter Contract Mapping** will not have any effect. The authentication context value for the adapter will still be the value returned from `lookupAuthN()`.

Another thing you might want to do is ensure that the authentication context value for the overall flow matches the value that was requested by the client or partner (through the `acr_values` request parameter, or the `RequestedAuthnContext` for SAML authentication requests). This can be done by adding a **Requested AuthN Context** selector to the authentication policy and configuring it to **Add or Update AuthN Context Attribute**. When you do this, the requested authentication context value is added to the attributes of the first authentication source encountered after the selector in the policy. The key for the added attribute is `SAML_AUTHN_CTX`. This attribute can then be added to the contract of that authentication source and mapped in the `SAML_AUTHN_CTX` of the APC to ensure that it becomes the authentication context value for the flow.

There's a last opportunity to override the authentication context value for a particular flow at the level of the service provider (SP) connection or the OIDC policy. For an SP connection, you can add `SAML_AUTHN_CTX` to the attribute contract and map the desired value into it. For an OIDC policy, you can add `acr` to the policy contract and map a value into it.

`SAML_AUTHN_INSTANT`

The `SAML_AUTHN_INSTANT` attribute is involved in determining the authentication instant value for the flow, which is mapped into the `auth_time` claim in the OIDC ID token or the `AuthnInstant` in the SAML assertion. This value is used by downstream applications to determine how recently the end user authenticated. Certain actions may be restricted if the authentication was not recent enough. The `AuthnInstant` is required in SAML assertions and will be set to the current time if a value can't be determined by the procedure below.

PingFederate determines the authentication instant value by iterating over each authentication source that was invoked in the flow. IdP adapters can return an authentication instant value by including the `org.sourceid.saml20.adapter.idp.authn.authnInst` attribute in the attribute map returned from `lookupAuthN()`.

For IdP connections, the authentication instant value comes from the `auth_time` claim in the ID token or the `AuthnInstant` value in the assertion.

When an authentication session exists for an authentication source and is used in the flow, the authentication instant value originally obtained from the IdP adapter or connection is used. If the source didn't return an authentication instant, the creation time of the session is used instead.

PingFederate iterates over the authentication sources that were invoked, pulling an authentication instant value from each one. The most recent authentication instant value returned by an authentication source becomes the authentication instant value for the overall flow.

Further control over the authentication instant value for the flow is available by adding `SAML_AUTHN_INSTANT` to the APC. Any value mapped into this attribute overrides the authentication instant value for the flow and will be passed to downstream applications through the ID token or assertion.

In some cases, you might want to populate the `SAML_AUTHN_INSTANT` in the APC with the value from a specific source. For SAML IdP connections, you can add `SAML_AUTHN_INSTANT` to the attribute contract. The corresponding attribute for OIDC IdP connections is `auth_time`. This will be automatically populated with the authentication instant value coming from that IdP connection and you can then map from that attribute into `SAML_AUTHN_INSTANT` in the APC.

For IdP adapters that return an authentication instant value, you can do the same thing by adding `org.sourceid.saml20.adapter.idp.authn.authnInst` to the IdP adapter contract.

> **Note**
>
> Mapping into the `org.sourceid.saml20.adapter.idp.authn.authnInst` attribute in the adapter's internal **Adapter Contract Mapping** will not have any effect. The authentication instant value for the adapter will still be the value returned from `lookupAuthN()` .

There's a last opportunity to override the authentication instant value for a particular flow at the level of the OIDC policy. You can add `auth_time` to the policy contract and map a value into it.

Adapter Mappings

Configuring adapter mappings allows administrators to map attributes from an authentication policy contract directly to a service provider (SP) adapter instance.

This allows the administrators to chain multiple authentication sources in an SP authentication policy, to build an authentication policy contract using attributes from authentication sources in the policy path, and to apply the authentication policy contract to the target application.

Configuring authentication policy adapter mappings

Authentication policy adapter mappings allow administrators to map attributes from an authentication policy contract directly to a service provider (SP) adapter instance.

Steps

1. Go to **Applications** > **Integration** > **Policy Contract Adapter Mappings**.
2. From the **Source Instance** list, select the applicable authentication policy contract from the .
3. From the **Target Instance** list, select the SP adapter instance integrated with your target application.
4. Click **Add Mapping**.
5. Follow the **Applications** > **Integration** > **Adapter-to-Adapter Mappings** wizard to create the mapping.
   1. **Optional:** On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source** to configure datastore queries to fulfill the SP adapter contract.

      Queries are executed in the order they are displayed on the **Attribute Sources & User Lookup** tab. Use the up and down arrows as needed to adjust the order.

      If a required attribute cannot be fulfilled, such as the user identifier of an adapter, the request fails. For more information, see Fulfillment by datastore queries.
   2. On the **Adapter Contract Fulfillment** tab, select a source and an attribute to fulfill the SP adapter contract.

      > **Note**
      >
      > Select **Authentication Policy Contract** from the **Source** list to map directly from the policy contract to the SP adapter contract or another choice to fulfill the SP adapter contract through datastore queries, dynamic texts, or results from OGNL expression

3. **Optional:** On the **Default Target URL** tab, specify a default target URL for this mapping configuration.

4. **Optional:** On the **Issuance Criteria** tab, configure conditions to be validated before issuing an SP adapter contract. For more information, see Define issuance criteria for adapter mapping.

5. On the **Adapter-to-Adapter Mapping Summary** tab, review the configuration and modify as needed. When complete, click **Done**.

6. On the **Adapter-to-Adapter Mappings** window, click **Save**.

Defining issuance criteria for adapter mapping

You can define issuance criteria for adapter mappings in PingFederate to conditionally approve or reject requests based on individual attributes. Satisfy criteria in order for PingFederate to move a request to the next phase.

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the Source list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Authentication Policy Contract	Select to evaluate attributes from the authentication policy contract.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. In the **Attribute Name** list, select the attribute to be evaluated.

3. In the **condition** list, select the comparison method.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**
   - **not equal to**
   - **not equal to (case insensitive)**
   - **not equal to DN**
   - **multi-value contains**
   - **multi-value contains (case insensitive)**
   - **multi-value contains DN**
   - **multi-value does not contain**
   - **multi-value does not contain (case insensitive)**
   - **multi-value does not contain DN**

   > **Note**
   >
   > The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

4. In the **Value** field, enter the comparison value.

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

5. In the **Error Result** field, enter a custom error message.

   To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

   If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. **Optional**: Repeat to add more criteria.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions

   1. Click **Show Advanced Criteria**
   2. In the **Expression** field, enter the required expressions.
   3. **Optional**: In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

- Token authorization
- Attribute mapping expressions
- Localizing messages for end users

Sessions

This topic describes the differences between application and authentication sessions.

Application sessions

Application sessions apply to PingFederate applications hosted on its user-facing endpoints, such as the profile management page and the grant management endpoints. When the inactivity threshold or the maximum lifetime is reached, PingFederate redirects previously authenticated users back to the authentication sources, identity provider (IdP) adapter instances or IdP connections, subject to the configuration of authentication sessions.

Authentication sessions

Authentication sessions control when PingFederate redirects previously authenticated users back to the authentication sources on subsequent requests for browser-based single sign-on (SSO) or PingFederate applications.

Authentication sessions typically wrap an adapter so that PingFederate creates the session when user authentication has succeeded. PingFederate invokes the adapter's authentication logic again only when the session reaches its limits. However, depending on the implementation, an adapter can be aware of an authentication session that wraps it and override this logic. In particular, PingFederate creates authentication sessions configured for an Identifier First Adapter instance only when the complete single sign-on (SSO) transaction has succeeded. This lets the adapter prompt the user for a different user identifier when a chained adapter authentication fails because, for example, there's a typo in the user identifier.

Session storage options

When authentication sessions are enabled, PingFederate maintains session data in memory.PingFederate also supports maintaining session data both in memory and on an external storage. This optional capability allows administrators to support use cases where a longer session duration or a greater resilience against restarts of PingFederate and browsers is desired. The retrieval and update operations are optimized to provide a fast and seamless user experience. For instance, a retrieval from the external storage is only required when an authentication session is not found in memory.

Note

Persistent authentication sessions require an external storage. For more information, see Defining a datastore for persistent authentication sessions.

Inactivity (idle) timeout and maximum lifetime

When authentication sessions are enabled, an authenticated user is not sent back to the authentication system as long as the user makes another request within the idle timeout window, 60 minutes by default. If the user makes another request within the idle timeout window, the authentication session is extended by the idle timeout value, another 60 minutes by default. For externally stored authentication sessions, this operation is optimized to only send updates to the external storage when the remaining idle timeout window is less than 75%.An authentication session can be repeatedly extended by multiple requests and remains valid until the maximum timeout value is reached, in which case the user will be redirected back to the authentication system.

Tip

The authentication system might or might not challenge the user to complete an authentication process based on its own session management policy or processing logic.

Configuration options

Administrators can enable authentication sessions for all authentication sources, with or without making the authentication sessions persistent, and with or without specifying overrides for selected authentication sources.Alternatively, administrators can enable authentication sessions for a few selected authentication sources, optionally with their own sets of overrides. The override options include:

- Disable or enable authentication sessions.
- Make authentication sessions persistent.
- Override the idle timeout, the maximum timeout, or both, in minutes, hours, or days.
- Enforce authentication requirement based on authentication context.

  Because sessions are tracked with their respective authentication context, administrators can optionally configure PingFederate to compare the requested authentication context found in the authentication request against the authentication context found in the session. If the values do not match, PingFederate redirects the user back to the authentication system.

Tracking options for logout

Administrators can optionally configure additional tracking options for logout to control whether PingFederate should leverage the single logout (SLO) application endpoints to terminate adapter sessions, add sessions to the session revocation list as users sign out, or do both. Publish revoked sessions to provide a secure SLO experience with PingAccess deployments.

Configuring tracking options for logout

You can configure PingFederate to track adapter sessions for logout.

About this task

An adapter session is a logout entry that, if tracked, ensures a logout request is sent to the adapter during single logout (SLO). Then the adapter can remove any session data that it is tracking for the user.

Steps

1. Go to **Authentication** > **Policies** > **Sessions**.
2. (Optional) Enable SLO for all adapter instances on a per-user basis by selecting the **Track Adapter Sessions for Logout** checkbox.

   When this checkbox is selected, an adapter session is tracked whenever an adapter is invoked during single sign-on (SSO). When this checkbox is cleared, the tracking of the adapter session depends on other factors, such as whether SLO is enabled on the partner connection involved in the SSO. This checkbox is cleared by default.

   > **Tip**
   >
   > This option is the simplest way to guarantee the cleanup of adapter sessions, but it does increase runtime memory usage. If you leave this box cleared, adapter sessions could persist after SLO.

3. (Optional) Add the associated sessions to the revocation list on logout by selecting the **Track Revoked Sessions on Logout** checkbox.

When selected, PingFederate always adds the associated sessions to the session revocation list as users sign off, even if an error occurs to the logout requests. This allows other systems, such as PingAccess, to query the validity of a given session at the Session Revocation API endpoint, `/pf-ws/rest/sessionMgmt/revokedSris`. This check box is selected by default for new installations.

> **Note**
>
> If your use cases involve OAuth requests, consider enabling the **Check session revocation status** option in the applicable Access Token Management instances so that the token validation process takes into account whether a session has been added to the revocation list.
> Learn more in Managing session validation settings.

4. (Optional) Change the number of minutes until the revoked sessions are removed from the revocation list for optimal performance by changing the value in the **Session Revocation Lifetime** field. You can enter an integer between 1 and 43200. The default value is 490 minutes.

> **Important**
>
> The **Session Revocation Lifetime** value should match or exceed the idle timeout value, or the maximum session lifetime value, of the authentication sources and the relying parties. For example, the default value of 490 minutes exceeds the global **Max Timeout** value for authentication sessions by 10 minutes to allow for clock skew among servers.

5. Click **Save**.

Related links

- OAuth client session management

Configuring application sessions

You can configure and override the default timeout limits for application sessions in the PingFederate administrative console.

Steps

1. Go to **Authentication** > **Policies** > **Sessions**.

2. **Optional:** On the **Sessions** window, override the default timeout values under **Application Sessions**.

Field	Description
Idle Timeout (Minute)	Modify the default inactivity timeout value in the **Idle Timeout (Minute)** field. You can enter an integer between `1` and `1576800`, representing a range of one minute to 1,095 days. You can also empty the value to indicate that the inactivity timeout value should match the maximum lifetime. The default value is `60` minutes.

Field	Description
Max Timeout (Minutes)	Modify the default maximum lifetime of an authentication session in the **Max Timeout (Minutes)** field. You can enter an integer between `1` and `1576800`, representing a range of one minute to 1,095 days. You can also empty the value to indicate that the authentication sessions do not expire until they are removed from the system. The value of the **Max Timeout (Minutes)** field cannot be less than that of the **Idle Timeout (Minute)** field. The default value is `480` minutes, which is 8 hours.

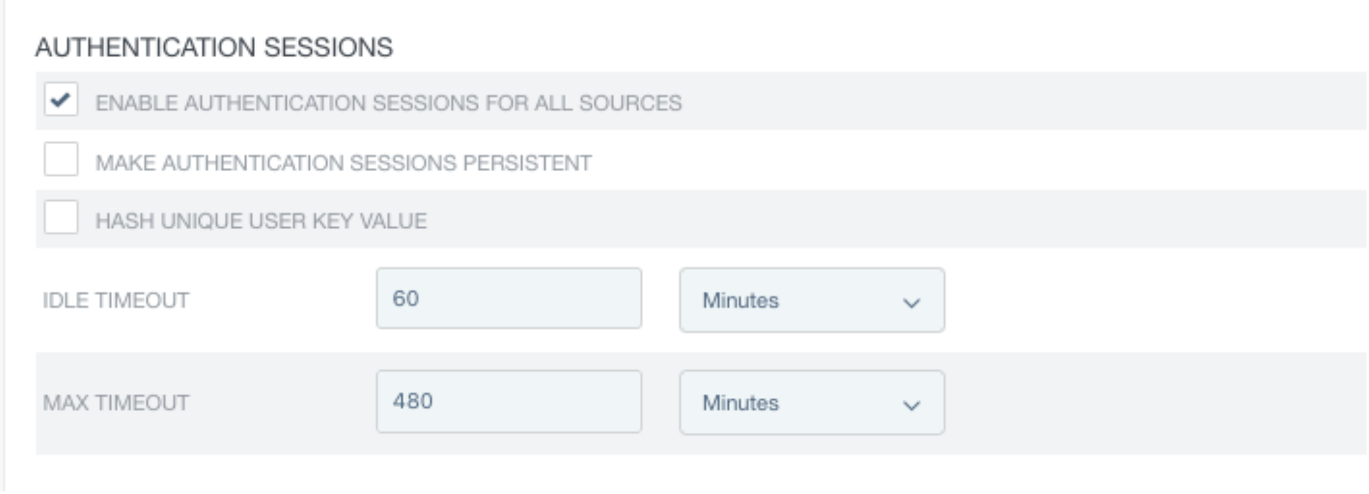
3. To keep your configuration, click **Save**.

Configuring authentication sessions

Use the **Sessions** window to configure and override the default timeout limits for authentication sessions.

Steps

1. Go to **Authentication** > **Policies** > **Sessions**.
2. **Optional:** In the **Sessions** window, configure the global policy and timeout settings under **Authentication Sessions**.



AUTHENTICATION SESSIONS

☑ ENABLE AUTHENTICATION SESSIONS FOR ALL SOURCES

□ MAKE AUTHENTICATION SESSIONS PERSISTENT

□ HASH UNIQUE USER KEY VALUE

IDLE TIMEOUT 60 Minutes

MAX TIMEOUT 480 Minutes

   1. Select the **Enable Sessions for All Authentication Sources** check box if PingFederate should track authentication sessions for all authentication sources. Clear this check box if you prefer to enable authentication sessions for only a few authentication sources or disable authentication sessions altogether. This check box is not selected by default.

> **Note**
>
> For any HTML Form Adapter instance that has been configured to allow users to indicate whether their device is shared or private, if a user signs on without selecting the **This is my device** check box on the login form, PingFederate removes authentication session information, if found, and does not store authentication sessions for the user.

2. If your use cases require longer sessions or greater resilience against restarts of PingFederate and browsers, select the **Make Authentication Sessions Persistent** check box.

   Selecting the check box causes the PF.PERSISTENT cookie to be set in the user's browser. By default, this cookie persists across browser restarts. To allow for very long sessions, the expiration period for the cookie defaults to 94608000 seconds, or 3 years. You can change this period in the **cookie-max-age** setting in the `persistent-session-cookie-config.xml` file. If you prefer to have the PF.PERSISTENT cookie cleared on browser exit, set `cookie-max-age` to `-1`. Regardless of the cookie's expiration period, PingFederate always enforces the configured session timeouts. However a user might lose their session earlier if the PF.PERSISTENT cookie expires or is removed by the browser.

   Persistent authentication sessions require an external storage.

   > **Tip**
   >
   > Your authentication session adapters should use either all persistent or all non-persistent sessions. Mixing usage can lead to undesired results, like non-persistent sessions being retained even after the session cookies clear on quitting the browser.

   > **Tip**
   >
   > You should use JWT-based persistent session cookies. You can configure this using the `cookie-format` parameter in the `persistent-session-cookie-config.xml` file. JWT-based cookies are the default setting.
   > JWT-based cookies improve security by allowing PingFederate to restore missing session information from the PF.PERSISTENT cookie rather than looking up session data from external persistent session storage.

   As of version 9.3, PingFederate alleviates DoS attacks by protecting the persistent session process. It does this by treating repeated persistent cookies that do not have a PF cookie as a replay if repeated in a specified time. This time is set to 300 seconds by default, and you can change it by modifying `EmptySessionReplayRetentionsSecs` in the `<pf-install>/server/default/data/config-store/org.sourceid.saml20.service.session.StoredSessionServiceImpl.xml` file.

   For example:

   - If a request arrives with a PF.PERSISTENT cookie and without a PF cookie, PingFederate starts counting the time set in `EmptySessionReplayRetentionsSecs`.
   - If another request arrives with the same PF.PERSISTENT cookie and without a PF cookie within the time specified in the configuration file, PingFederate treats it as a replayed request and does not perform a database lookup.

   You can disable this behavior by setting `EmptySessionReplayRetentionsSecs` to `0`.

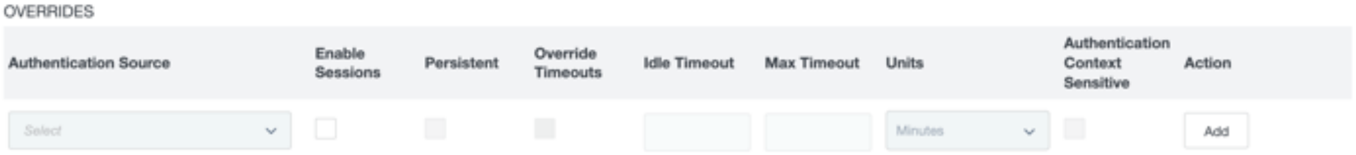
3. Select the **Hash Unique User Key Value** check box if you want the unique user key to be hashed using SHA-256. When this option is enabled, PingFederate associates this hashed value with the particular user's authentication sessions.

The hashed value is used for features related to unique user keys; for example, the HTML Form Adapter's **Revoke Sessions After Password Change or Reset** option (for more information, see Configuring an HTML Form Adapter instance). The hashed value will be visible in server and audit logs, and in session storage if **Make Authentication Sessions Persistent** is enabled.

1. **Optional:** Override the default timeout values for all authentication sources.

Field	Description
Idle Timeout	Modify the default inactivity timeout value in the **Idle Timeout** field and select a unit of measurement from the list. You can enter an integer that represents a time period between 1 minute and 1,095 days. You can also empty the value to indicate that the inactivity timeout value should match the maximum lifetime. The default inactivity timeout value is `60` minutes.
Max Timeout	Modify the default maximum lifetime of an authentication session in the **Max Timeout** field and select a unit of measurement from the list. You can enter an integer that represents a time period between 1 minute and 1,095 days. You can also empty the value to indicate that the authentication sessions do not expire until they are removed from the system. The value of the **Max Timeout** field cannot be less than that of the **Idle Timeout** field. The default maximum timeout value is `480` minutes (eight hours).

3. **Optional:** Configure policy and settings for individual authentication sources under **Overrides**.



OVERRIDES

Authentication Source	Enable Sessions	Persistent	Override Timeouts	Idle Timeout	Max Timeout	Units	Authentication Context Sensitive	Action
Select	□	□	□			Minutes	□	Add

1. From the **Authentication Source** list, select an identity provider (IdP) adapter instance or an IdP connection.

2. Configure individual policy for the selected authentication source as follows.

Global policy (underAuthentication Sessions)	Individual policy (under Overrides)
The **Enable Sessions for All Authentication Sources** check box is not selected. Authentication-session tracking is not enabled for all authentication sources.	Select the **Enable Sessions** check box to enable authentication-session tracking for the selected authentication source.
The **Enable Sessions for All Authentication Sources** check box is selected. Authentication-session tracking is enabled for all authentication sources.	Clear the **Enable Sessions** check box to disable authentication-session tracking for the selected authentication source. Select the **Enable Sessions** check box for the purpose of overriding other authentication-session settings for the selected authentication source.

> **Note**
>
> The **Enable Sessions** check box is not selected by default.
> For any HTML Form Adapter instance that has been configured to allow users to indicate whether their device is shared or private, if a user signs on without selecting the **This is my device** check box on the login form, PingFederate removes authentication session information, if found, and does not store authentication sessions for the user.

3. Select the **Persistent** check box if your use cases require a longer session duration or a greater resilience against restarts of PingFederate and browsers.

   Available and applicable only if the **Enable Sessions** check box is selected. The **Persistent** check box is not selected by default.

   > **Note**
   >
   > Persistent authentication sessions require an external storage.

   > **Note**
   >
   > Notes under step 2b apply here as well.

4. If authentication-session tracking is enabled for the selected authentication source and if you want to configure specific timeout values, select the **Override Timeouts** check box and configure timeout settings.

Field	Description
Idle Timeout	You can enter an integer that represents a time period between 1 minute and 1,095 days. You can also empty the value to indicate that the inactivity timeout value should match the maximum lifetime. This field has no default value.
Max Timeout	You can enter an integer that represents a time period between 1 minute and 1,095 days. You can also empty the value to indicate that the authentication sessions do not expire until they are removed from the system. The value of the **Max Timeout** field cannot be less than that of the **Idle Timeout** field. This field has no default value.
Unit	Select from the list the unit of measurement for both the **Idle Timeout** and **Max Timeout** fields. The default selection is **Minutes**.

5. If authentication-session tracking is enabled for the selected authentication source and if you want to enforce authentication requirement based on the authentication context for the selected authentication source, select the **Authentication Context Sensitive** check box. This check box is not selected by default.

6. Click **Add**.

7. Repeat these steps to configure individual policy and settings for additional authentication sources.

Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

4. To save your configuration changes, click **Save**.

Result

When PingFederate authentication sessions are enabled, you can configure session-validation options for your OAuth use cases. These optional settings enable you to conjoin the validity of access tokens and the authentication sessions of the users. For more information, see Managing session validation settings.

Related links

- Defining a datastore for persistent authentication sessions
- Configuring an HTML Form Adapter instance

Bundled adapters

PingFederate comes bundled with a set of adapters for integration and customizable configuration, depending on your service needs.

PingFederate's bundled set of adapters:

- Composite Adapter
- HTML Form IdP Adapter
- HTTP Basic IdP Adapter
- Identifier First Adapter
- Kerberos Adapter
- OpenToken IdP Adapter
- OpenToken SP Adapter
- Passthrough IdP Adapter
- PingID Adapter
- PingOne MFA Adapter
- PingOne Protect Adapter
- PingOne Verify Adapter
- PingOne DaVinci Adapter
- Reference ID IdP Adapter
- Reference ID SP Adapter
- X.509 Certificate IdP Adapter

Composite Adapter

PingFederate’s Composite Adapter allows adapter chaining for single sign-on (SSO) requests that best fit your user or organization’s needs.

For an Identity Provider (IdP), PingFederate includes a Composite Adapter that allows an administrator to chain the selection of available adapter instances for a connection. At runtime, adapter chaining means that SSO requests are passed sequentially through each adapter instance specified until one or more authentication results are found for the user.

Adapter chaining can be used to choose an adapter instance based on the method by which a user authenticated, or to integrate an organization’s multi-factor authentication requirement.

> **Tip**
>
> For complex authentication requirements, consider implementing authentication policies. Go to **Authentication > Policies**. From the **Policies** window, make changes on the **Policies** tab.

Configuring a Composite Adapter instance

Configuring a Composite Adapter instance allows you to enable adapter chaining for a user authentication connection.

About this task

Configure an instance of a Composite Adapter in PingFederate using the administrative console.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. On the **IdP Adapters** window, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.
    3. **Optional**: In the **Parent Instance list**, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.
4. On the **IdP Adapter** tab, configure your Composite Adapter instance as follows:
    1. Click **Add a new row to 'Adapters'**.
    2. From the **Adapter Instance** list, select an IdP adapter instance. Click **Update**.

       For more information, see the **Description** column in each configuration section and the following table.

PingFederate's column names and descriptions for creating an adapter instance

Column	Description
Policy (Required)	**Required** (the default) indicates authentication through this adapter instance is needed to continue SSO processing and to invoke any remaining instances in the chain. If you are integrating multi-factor authentication, use this policy for each instance. The Composite Adapter instance returns an error when the authenticate attempt against a required adapter instance fails. **Sufficient** indicates that authentication through this adapter instance is enough to satisfy requirements, along with any required instances that have already been selected. Any subsequent configured instances in the chain are not invoked. **Important** For the sufficient policy to work correctly, the adapter must return control to PingFederate after any kind of a failure.
AuthN Context Weight (Required)	If more than one adapter instance in the chain is capable of returning an authentication context, this relative weight is used to determine which value is included in the assertion, unless the value is overridden under **AuthN Context Override**. If weights are the same for two or more contexts, the first one processed is included in the assertion. The default value is `3` .
AuthN Context Override	If provided, this value overrides the authentication context value that might be returned from the adapter instance. The value in this field can be sent in the assertion if the associated adapter instance is invoked. If weights are the same for two or more contexts, the first one processed is included in the assertion.

3. Add at least one more adapter instance and configure its **Policy**, **AuthN Context Weight**, and **AuthN Context Override** settings.

   Repeat this step to add more adapter instances as needed.

   At runtime adapter chaining is sequential, starting at the top of the list.

   > **Important**
   >
   > You can configure several types of adapters — for example, the IWA IdP Adapter — to direct end users to an error page if authentication fails for any reason, which will halt further progress through a composite-adapter chain. For such adapter instances, ensure the **Error URL** option is not used in the instance configuration when continuation through an adapter-chaining sequence is required.

4. **Optional:** In the **Action** column, manage the selected adapter instances.

5. Configure **Input User ID Mapping**.

If you have configured any IdP Adapter developed using the `IdpAuthenticationAdapterV2` interface from the PingFederate SDK, including the HTML Form Adapter, the **Input User ID Mapping** section appears. Additionally, some IdP adapters, such as the PingID Adapter and the separately available Symantec VIP Adapter, require a user ID to be passed in from an earlier-authentication step to perform multifactor authentication. If so, an administrator must specify the attribute containing the unique ID on this window. For example, to pre-populate the username of an HTML Form Adapter instance with an attribute from an earlier authentication source in the previous steps:

1. Click **Add a new row to 'Input User ID Mapping'**.
2. From the **Target Adapter** list, select the HTML Form Adapter instance.
3. From the **User ID Selection** list, select a source attribute.
4. Click **Update**.

> ⓘ **Note**
>
> For OAuth use cases, entries in the **Input User ID Mapping** section might override the `login_hint` parameter value provided by the OAuth clients when they submit their requests to the `/as/authorization.oauth2` authorization endpoint.

> **Tip**
>
> By default, the HTML Form Adapter does not allow the users to change the username if it is configured to be pre-populated with an attribute from another authentication source. You can override this restriction by enabling the **Allow Username Edit** option on a per-adapter instance.

6. Configure **Attribute Name Synonyms**.

If any attributes are logically equivalent across two adapter instances but have different names, click **Add a new row to 'Attribute Name Synonyms'**. Select attributes from the **Name** and **Synonym** lists to create a mapping between them.

The attribute name under **Synonym** and its value are used in the SAML assertion when the two values returned from each adapter are identical. If returned values are different, both values are sent for the synonym.

> ⓘ **Note**
>
> Without this configuration to identify synonymous attribute names, both names and their values are sent in the SAML assertion.

7. Define the order in which different values are returned for the same attribute name.

For attributes of the same name configured in different adapter instances, you can change the order of returned values when the values are different. (Values are merged if they are the same.)

By default, the **Add to Back** value for an attribute name configured in the first instance is returned first and also listed first in the resulting SAML assertion. Then any different value from the same attribute name in a subsequently invoked instance is appended.

The order might not matter for many attributes, but in the case of the SAML-subject attribute, only the first value in the SAML assertion can be used for an SP connection partner under normal circumstances. Click **Add to Front** to reverse the default order, if needed.

5. On the **Extended Contract** tab, click **Add** to add attributes to be returned from each adapter instance configured on the previous window.

   > **Note**
   >
   > Attributes must correspond exactly to any or all of the attribute names listed on the Adapter Attribute tab for each configured adapter instance.

6. On the **Adapter Attributes** tab, do the following:

   1. **Optional**: From the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

      The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

      > **Note**
      >
      > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

      > **Important**
      >
      > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

   2. Select the check box under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

      This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

      > **Note**
      >
      > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

   3. Select the check box under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

      > **Note**
      >
      > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

   4. Select the **Mask all OGNL-expression generated log values** check box if OGNL expressions might be used to map derived values into outgoing assertions and you want those values masked.

7. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

    - Configure one or more data sources for datastore queries.
    - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
    - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

8. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.
9. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

    If you want to exit without saving the configuration, click **Cancel**.

HTML Form Adapter

The HTML Form Adapter supports user authentication when it occurs outside of the PingFederate server through an application or the authentication module of an identity access management (IAM) system that leverages multiple user repositories and a password credential validator (PCV) instance.

Initial user authentication is normally handled outside of the PingFederate server using an application or an IAM system authentication module. Adapters or agents from PingFederate integration kits are typically used to integrate with these local authentication mechanisms.

PingFederate packages an HTML Form Adapter that delegates user authentication to a PCV, such as an LDAP Username PCV. The adapter validates credentials against a user repository through a PCV instance.

To validate against multiple user repositories, you can add multiple PCV instances to an instance of the HTML Form Adapter. In this case, if a PCV instance fails to validate the user credentials, PingFederate uses the next PCV instance.

When PingFederate receives an authentication request and the use case involves an HTML Form Adapter instance, PingFederate invokes the adapter if it does not find a valid authentication session. If the HTML Form Adapter does not find a valid adapter session, it displays a sign-on page and prompts the user for credentials.

If you configure and enable customer IAM, users can optionally register local accounts or sign on using third-party identity providers (IdPs). If a user chooses to sign on using local accounts, the credentials are validated using the designated PCV instance or instances. If validated, PingFederate generates the requested single sign-on (SSO) token or moves the request to the next checkpoint if authentication policies are involved.

In terms of the sign-on experience, the HTML Form Adapter lets you:

- Use different customizable and localizable template files.
- Define a logout path or a logout redirect page.
- Notify users with password expiry information.
- Let users change or reset their network passwords, or redirect users to a company-hosted password management system.
- Enable self-service password reset, account unlock, and username recovery.

You can configure all capabilities on a per-adapter instance basis.

PingFederate also tracks sign-on attempts per adapter instance, which adds a layer of protection against brute force and dictionary attacks. When the **Challenge Retries** threshold is reached, PingFederate locks out the user for a period of time. The default value for the **Challenge Retries** setting is 3. If a higher value is preferred, consider reviewing the account lockout policy of the user repository first. For example, if the account lockout threshold is set to 5 on the target directory server and the **Challenge Retries** setting is also set to 5 or higher, the fifth sign-on attempt could lock the user accounts on the directory server. The lockout period is controlled by the Account Locking Service.

> **Note**
>
> The HTML Form Adapter considers a password consisting only of spaces as empty and does not refer to the underlying LDAP for authentication. As a result, while passwords containing only spaces will trigger PingFederate's Account Locking Service, the service will not lock the account at the LDAP level even if the user exceeds the LDAP limit for sign-on attempts. If the LDAP service locks the account due to too many failed sign-on attempts with passwords that contain non-space characters, and a user attempts to log in using a password consisting only of spaces after the Account Locking Service's lockout period has ended, PingFederate will report the situation as an invalid password, rather than a locked account.

This adapter does not provide an authentication context. For SAML connections, PingFederate sets the authentication context as follows:

- `urn:oasis:names:tc:SAML:1.0:am:unspecified` for SAML 1.x
- `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` for SAML 2.0

PingFederate can override the authentication context with either an instance of the Requested AuthN Context Authentication Selector or the `SAML_AUTHN_CTX` attribute in the SAML attribute contract. The latter takes precedence.

> **Note**
>
> The HTML Form Adapter is authentication API-capable. The PingFederate authentication API is a JSON-based API that enables end-user interactions, such as credential prompts, to be handled by an external web application. This API does so by providing access to the current state of the flow as an end user steps through a PingFederate authentication policy.
> For more information, see Authentication applications and the authentication API.

Configuring an HTML Form Adapter instance

In the **IdP Adapters** page, configure an HTML Form Adapter instance to validate a user authentication session with a password credential validator (PCV) when your initial authentication needs to integrate with an external application or an identity management system (IdM) authentication module.

Steps

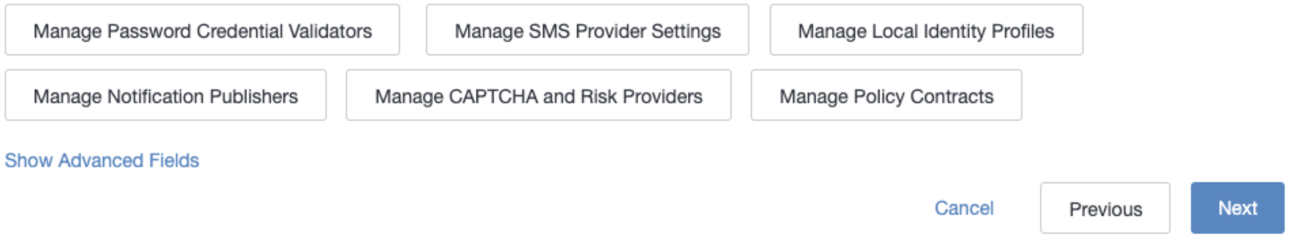
1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. In the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.

3. Optional: In the **Parent Instance** list, select an existing type.

   If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

4. On the **IdP Adapter** tab, configure your HTML Form Adapter instance as follows:

   1. If you have not yet defined the desired PCV instance, click **Manage Password Credential Validators**.


Manage Password Credential Validators
Manage SMS Provider Settings
Manage Local Identity Profiles
Manage Notification Publishers
Manage CAPTCHA and Risk Providers
Manage Policy Contracts
Show Advanced Fields
Cancel
Previous
Next

   2. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.

   3. From the **Password Credential Validator Instance** list, select a PCV instance. Click **Update**.

      Add as many validators as necessary. Use the up and down arrows to adjust the order in which you want PingFederate to attempt credential authentication. If the first mechanism fails to validate the credentials, PingFederate moves sequentially through the list until credential validation succeeds. If none of the PCV instances can authenticate the user’s credentials, and the challenge retries maximum has been reached, the process fails.

      > **Note**
      >
      > If usernames overlap across multiple PCV instances, this failover setup could lock out those accounts in their source locations.

   4. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.

   5. Enter values for the adapter configuration, as described below.

Field	Description
Challenge Retries (Required)	The account lockout threshold for this adapter instance. When the number of login failures reaches this threshold, the user is locked out for a period time. The default value is `3`.

Field	Description
Session State	Determines whether this HTML Form Adapter instance maintains adapter sessions and shares adapter sessions with other HTML Form Adapter instances. ***Globally*** Adapter sessions from this HTML Form Adapter instance are shared among other HTML Form Adapter instances that use the same **Session State** field value **Globally**. ***Per Adapter*** HTML Form Adapter maintains adapter sessions on a per-instance basis. Sessions from this HTML Form Adapter instance are not shared with other HTML Form Adapter instances. ***None*** This HTML Form Adapter does not maintain adapter sessions for this HTML Form Adapter instance. **Note** To enable PingFederate authentication sessions globally or individually for this adapter instance, select **None**. Learn more about PingFederate authentication sessions in Sessions and Configuring authentication sessions. The default selection is **None**.
Session Timeout	The number of idle minutes before an HTML Form Adapter session times out based on inactivity. If left blank, the lifetime falls back on the **Session Max Timeout** field value. Ignored if **None** is selected for the **Session State** field. Applicable only when the **Session State** field is set to **Globally** or **Per Adapter**. **Tip** When you enable PingFederate authentication sessions globally or individually for this adapter instance, you can configure the **Idle Timeout** setting for the same purpose. Learn more in Configuring authentication sessions. The default value is `60` minutes.

Field	Description
Session Max Timeout	The maximum lifetime, in minutes, before an HTML Form Adapter session expires regardless of whether the **Session Timeout** field value has been reached. Ignored if **None** is selected for the **Session State** field. Applicable only when the **Session State** field is set to **Globally** or **Per Adapter**. **Tip** When you enable PingFederate authentication sessions globally or individually for this adapter instance, you can configure the **Max Timeout** setting for the same purpose. Learn more in Configuring authentication sessions. The default value is `480` minutes, which translates to 8 hours. **Note** This setting sets a maximum lifetime, subject to inactivity timeout. Consider the following examples: ■ A user initiated an single sign-on (SSO) request at 9 a.m. and has not made another SSO request since then. At 10 a.m., the HTML Form Adapter session times out based on inactivity based on the default **Session Timeout** field value of 60 minutes. ■ Another user initiated an SSO request at 9 a.m. and has been making SSO requests every hour at least once. This HTML Form Adapter session does not time out because the user has been actively making SSO requests; however, the HTML Form Adapter session does expire at 5 p.m. based on the default **Session Max Timeout** default value of 8 hours. ■ If you leave both the **Session Max Timeout** and **Session Timeout** fields blank, HTML Form Adapter sessions do not expire until PingFederate restarts or the HTML Form Adapter sessions are cleaned up by another means. ■ If you leave the **Session Max Timeout** field blank but set a value for the **Session Timeout** field, HTML Form Adapter sessions do not expire until they time out based on inactivity. **Tip** Session information is stored in the PF cookie. By default, the PF cookie is a session cookie and is typically removed when the user closes the browser. You can optionally extend the lifetime of the PF cookie by editing the `session-cookie-config.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory. Learn more in Extending the lifetime of the PingFederate cookie. Alternatively, you can enable PingFederate authentication sessions, store the authentication sessions externally, and leverage them as users request protected resources after restarting their browsers. Learn more in Sessions.

Field	Description
Allow Password Changes	Enables or disables the ability for users to change their network password using this adapter instance as they initiate SSO requests and are prompted to enter their username and password. As needed, you can also provide your users the Change Password endpoint shown on the **Summary** window. The Change Password endpoint allows users to change their password without submitting SSO requests. For more information, see the `/ext/pwdchange/Identify` section. ⓘ **Note** The LDAP Username PCV and the PingOne PCV are currently the only PCVs bundled with PingFederate that support the change password feature. **Important** When connecting to an Active Directory (AD) server, you must secure the datastore connection using LDAPS. AD requires this level of security to allow password changes. This checkbox is not selected by default.
Password Management System	The URL for redirecting users to a company-specific password management system to change their password. This field has no default value.
Enable 'Remember My Username'	Allows users to store their username as a cookie when authenticating with this adapter. Once stored, the username in the login form is pre-populated for subsequent transactions. Select the checkbox to enable the cookie functionality. ⓘ **Note** This option is hidden when users authenticate through a Composite Adapter instance that chains this adapter behind another authentication source with an **Input User ID Mapping** configuration and the **Allow Username Edits** checkbox is not selected. This checkbox is not selected by default.
Enable 'This is My Device'	Allows users to indicate whether their device is shared or private by selecting the **This is my device** checkbox on the login form. In this mode, PingFederate authentication sessions, if enabled, are not stored unless the user indicates the device is private. Learn more about PingFederate authentication sessions in Sessions. This checkbox is not selected by default. ⓘ **Note** Adapter session tracking, if enabled by setting the **Session State** field to **Globally** or **Per Adapter**, is not affected by this configuration and the user’s selection.

Field	Description
Change Password Policy Contract	Select an authentication policy contract to enforce strong authentication requirements, such as multi-factor authentication through PingID, before letting their users change their passwords. This is similar to using a **Password Reset Policy Contract**. The field is empty by default.
Change Password Notification	When selected, a notification is generated for the user who has successfully changed the password through the HTML Form Adapter. The destination is the user's email address, specifically the mail attribute value returned by the LDAP Username PCV instance. **Note** This option requires the selection of the **Allow Password Changes** checkbox and a notification publisher instance. If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports this notification feature. This checkbox is not selected by default.

Field	Description

Show Password Expiring Warning	When selected, the HTML Form Adapter displays a warning to an authenticated user if the password associated with the account is about to expire soon. The message provides the number of days until the expiry of the current password and the options to change the password immediately or to snooze the message. Both the threshold and the snooze interval are configurable in the **Advanced fields** section. The default values are 7 days and 24 hours, respectively.
	This option requires the selection of the **Allow Password Changes** checkbox. Both checkboxes are cleared by default.
	Note The LDAP Username PCV is the only PCV bundled with PingFederate that supports the password expiring warning feature.
	To use this option with PingDirectory:
	■ Use PingDirectory's **`dsconfig`** utility and a command like the following to enable the `ds-pwp-state-json` virtual attribute for users with the object class `person`:
	`dsconfig set-virtual-attribute-prop \`
	`   --name "Password Policy State JSON" \`
	`   --set enabled:true \`
	`   --set require-explicit-request-by-name:true \`
	`   --set "filter:(objectClass=person)"`
	Note The `ds-pwp-state-json` virtual attribute is required for the expiration warning page to function as expected. This attribute was added in PingDirectory 8.2 and cannot be added manually to earlier versions.
	■ In PingDirectory, set the `return-password-expiration-controls` setting in the user's password policy to `always` using a command like the following:
	`dsconfig set-password-policy-prop \`
	`   --policy-name "Default Password Policy" \`
	`   --set return-password-expiration-controls:always`
	■ In the LDAP Username PCV, enable the **Expect Password Expired Control** setting. Learn more in Configuring the LDAP Username Password Credential Validator.

Password Reset Type	Select one of the following methods for self-service password reset (SSPR). ***Authentication Policy*** Based on the policy contract selected from the **Password Reset Policy Contract** list, PingFederate finds the applicable authentication policy to handle self-service password reset requests. If the users are able to fulfill the authentication requirements as specified by the policy, PingFederate allows the users to reset their password. ***Email One-Time Link*** Users receive a notification with a URL to reset their password.If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. ***Email One-Time Password*** Users receive a notification with a one-time passcode (OTP) to reset their password. If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. ***PingID*** Users are prompted to follow the PingID authentication flow to reset their password. Ensure the **PingID Username Attribute** field in the selected LDAP Username PCV instance is configured. Otherwise, users will not be able to reset their password. You must also download the settings file from the PingID admin portal and upload the file to the **PingID Properties** advanced field. **Important** Do not use a method that is already part of a multi-factor authentication (MFA) policy that includes a password challenge because that would indirectly reduce that authentication policy to a single factor. For example, if users normally authenticate with a password challenge and then PingID, the self-service password reset method should not be PingID. Instead, choose the **Authentication Policy** option, select a policy contract in the **Password Reset Policy Contract** list, and configure an authentication policy for self-service password reset. ***Text Message*** Users receive a text message notification with an OTP to reset their password. Ensure the **SMS Attribute** field in the selected LDAP Username PCV instance is configured. Otherwise, users will not receive text message notification for password reset.If you have not yet configured SMS provider settings in PingFederate, click **Manage SMS Provider Settings**. ***None*** Users cannot reset password through this HTML Form Adapter instance. The default selection is **None**. When you make a selection other than **None**, as users initiate SSO requests and are prompted to enter their username and password, users have the option to reset their password.

Field	Description
	As needed, you can also provide your users the Account Recovery endpoint shown on the **Summary** tab. The Account Recovery endpoint allows users to change their password without submitting SSO requests. Learn more in the `/ext/pwdreset/Identify` section in IdP endpoints. ⓘ **Note** To enable password reset, you must also select the **Allow Password Changes** checkbox. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports SSPR. If a notification publisher instance is configured, PingFederate generates a notification for the user who has successfully reset the password through the HTML Form Adapter. The destination is the user's email address, specifically the value of the attribute defined by the **Mail Attribute** setting in the LDAP Username PCV instance. **Important** When connecting to PingDirectory, Oracle Unified Directory, or Oracle Directory Server, configure proxied authorization for the service account on the directory server. Learn more in Proxied authorization.
Password Reset Policy Contract	If you use an authentication policy to handle SSPR requests, you must select a policy contract here. This policy contract doesn't require any extended attributes because PingFederate uses this policy only to find the applicable authentication policies for password resets. **Important** You must use a policy contract dedicated only to password reset. You can't use this policy contract for single sign-on (SSO) anywhere else. To define a policy contract solely for password reset, click **Manage Policy Contracts**. An authentication policy that uses this contract allows users to reset their password. Ensure the policy uses strong authentication methods to securely identify the user who initiated the password reset operation. Map the incoming user ID for adapters in the policy to **Requested User** and confirm that adapters will only return success when this user is the one authenticating. Developing IdP adapters has guidelines on designing adapters for use in password reset or password change authentication policies.

Field	Description
Revoke sessions after password change or reset	Revokes a user's authentication sessions in other browsers after a password change or reset is completed by this adapter. This option relies on selecting a unique user key attribute for this adapter (learn more in Setting pseudonym and masking options). To enable this option, you must also enable the **Allow Password Changes** option, or set the **Password Reset Type** option to something other than **None**. ⓘ **Note** This revocation capability is not supported if the IdP session registry is configured with the Directed Clustering - Subclusters state management architecture. Learn more in IdP Session Registry Service and Defining subclusters.
Account Unlock	Enables or disables the ability for users to unlock their account using this adapter instance as they initiate SSO requests and are prompted to enter their username and password. As needed, you can also provide your users the Account Recovery endpoint shown on the **Summary** tab. The Account Recovery endpoint allows users to unlock their account without submitting SSO requests. Learn more in the `/ext/pwdreset/Identify` section of IdP endpoints. ⓘ **Note** You must also select a **Password Reset Type** value other than **None** and the **Allow Password Changes** checkbox as well because the initiating user must prove ownership of the account through the password reset flow. Unlike self-service password reset self-service password reset (SSPR), when users succeed in proving account ownership, they are allowed to retain their current password or to reset their password as needed. Furthermore, self-service account unlock is only compatible with PingDirectory and Microsoft AD. If the underlying datastore is connected to an Oracle Unified Directory or Oracle Directory Server, users can only unlock their account by changing their current password through the password reset flow. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports self-service account unlock. This checkbox is not selected by default.
Local Identity Profile	Select a local identity profile to offer users the options to authenticate through third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page. There is no default selection.
Notification Publisher	If this adapter instance is configured with self-service account management capabilities, select a notification publisher instance from the list. Based on selected notification publisher instance configuration, PingFederate generates the required notification messages. If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

Field	Description
Enable Username Recovery	Enables or disables the ability for users to recover their username when using this adapter instance as they initiate SSO requests and are prompted to enter their username and password. As needed, you can also provide your users the Username Recovery endpoint shown on the **Summary** tab. The Username Recovery endpoint allows users to recover their username without submitting SSO requests. Learn more in the `/ext/idrecovery/Recover` section of IdP endpoints. ⓘ **Note** This capability requires a notification publisher instance. If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports self-service username recovery. For each username recovery request, if PingFederate can locate the user record using the email address provided by the user and other requirements are met, PingFederate generates a notification containing the recovered username. The destination is the email address provided by the user. This checkbox is not selected by default.

6. (Optional) Click **Show Advanced Fields** to review or modify default values.

7. If you have chosen **Text Message** as the password reset type, click **Manage SMS Provider Settings** at the bottom of the page to configure the SMS provider through which PingFederate can send text message notifications to the users.

5. On the **Extended Contract** tab, configure additional attributes for this adapter instance as needed.

The HTML Form Adapter contract includes two core attributes: `username` and `policy.action`. At runtime, PingFederate fulfills the `policy.action` core attribute as described in the following table.

Local identity profile	Runtime fulfillment
A selection is made.	If the local identity profile is configured with one or more authentication sources, and if the user chooses to register or authenticate with one of them, PingFederate sets the value to that authentication source. This design allows you to create rules in your authentication policies and form different policy paths for each authentication source. Learn more in Enabling third-party identity providers. Whether or not the local identity profile is configured with any authentication sources, if the user chooses to register directly by clicking on the **Register now** link, PingFederate sets the value to `identity.registration`. This fulfillment allows you to create rules to differentiate authentication requirements from the registration flow. Learn more in Creating advanced registration mapping.
No selection is made.	The `policy.action` attribute is not fulfilled.

6. On the **Adapter Attributes** tab, do the following:

    1. (Optional) From the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

        The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

        > **Note**
        >
        > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's PCV to trim spaces.

        > **Important**
        >
        > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

    2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

        This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

        > **Note**
        >
        > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

    3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

        > **Note**
        >
        > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

    4. Select the **Mask all OGNL-expression generated log values** checkbox if OGNL expressions might be used to map derived values into outgoing assertions and you want those values masked.

7. (Optional) On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

    1. Configure one or more data sources for datastore queries. Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.

    2. Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

8. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

9. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

If you want to exit without saving the configuration, click **Cancel**.

HTML Form Adapter advanced fields

When configuring an HTML Form Adapter, you can use the advanced fields at the bottom of the **IdP Adapter** tab in the **Create Adapter Instance** window.

Advanced fields for setting password credentials and changes

Property	Description
Login Template (Required)	The HTML core template to prompt the users for their credentials. PingFederate allows each configured adapter instance to use a different login page template. The default template file is `html.form.login.template.html`. Unless otherwise stated, all template files are located in the `<pf_install>/pingfederate/server/default/conf/template` directory.
Logout Path	Any path in the format indicated. Setting a path invokes adapter logout functionality that is normally invoked during SAML 2.0 single logout (SLO) processing. The resulting logout path is `/ext/<Logout Path>`. The logout path extends from the base URL. If virtual host names are configured, the logout path is accessible at those locations as well. Available primarily for use cases where the partner software as a service (SaaS) providers who do not support SAML SLO but want the users' IdP-initiated SSO sessions to end after logging out of the SaaS services. For these use cases, the SaaS providers could redirect the users to the logout URL after the users sign out of their platforms. ⓘ **Note** If specified, the path must be unique across all HTML Form Adapter instances, including child instances. This field has no default value.
Logout Redirect	The landing page at the service provider (SP) after successful identity provider (IdP) logout, applicable only when the **Logout Path** field is configured. This field has no default value.
Logout Template	The HTML template displayed when a user has successfully logged out in a configuration where the **Logout Path** field is configured, but the **Logout Redirect** field is not. The default template file is `idp.logout.success.page.template.html`.
Change Password Template	The HTML core template to prompt the users to change their password. PingFederate allows each configured adapter instance to use a different change password template. The default template file is `html.form.change.password.template.html`.

Property	Description
Change Password Message Template	The HTML template to be displayed when a user has successfully changed the password through the HTML Form Adapter. The default template file is `html.form.message.template.html`.
Password Management System Message Template	The HTML template notifies the users that they are being redirected to a password management system to change their password. The default template file is `html.form.message.template.html`.
Password Timeout Update	The time, in minutes, a user has for a password change session on the **Change Password** and **Reset your password** pages. By default, the value is 30 minutes and this feature is enabled. If the field is left blank, this feature is disabled. ⓘ **Note** • To use this feature for the self-service password reset (SSPR) flow, you must select a **Password Reset Type** other than **None**. • To use this feature for the change password with an authentication policy flow, you must select a policy contract for the **Change Password Policy Contract** and enable **Allow Password Changes**.
Change Password Email Template	The HTML email template PingFederate uses to generate the email message to notify the user that the password has been changed or reset successfully through the HTML Form Adapter. The default template file is `message-template-end-user-password-change.html`, located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory. Applicable only if an instance of the SMTP Notification Publisher is selected in the **Notification Publisher** list.
Expiring Password Warning Template	The HTML core template to warn the users about the approaching password expiration day. The default template file is `html.form.password.expiring.notification.template.html`.
Threshold for Expiring Password Warning	The threshold, in days, to start warning the user about the approaching the password expiration day. The default value is `7` days.
Snooze Interval for Expiring Password Warning	The amount of time, in hours, to delay the next warning after the user has chosen to change the password later. The default value is `24` hours.
Require Re-authentication for Expiring Password Flow	Requires a user to sign on again after changing their password if they initiated the change on the password expiring warning. By default, this feature is disabled. Applicable only when the **Show Password Expiry Warning** is enabled.

Property	Description
Require Re-authentication for Change Password Flow	Requires a user to sign on again with their new password after completing a successful change password flow. By default, this feature is disabled.
Require Re-authentication for Password Reset Flow	Requires a user to sign on again with their new password after completing a successful password reset or account unlock flow. By default, this feature is disabled.
Login Challenge Template	The HTML core template to be displayed as the second step during a strong authentication. It is used to prompt the user to answer a challenge question after the first-factor login. The RADIUS Username password credential validator (PCV) is an example of where it could be used. The default template file is `html.form.login.challenge.template.html`.
'Remember My Username' Lifetime	The number of days the cookie remains valid. Enter the number of days you want the username remembered in a cookie. The cookie lifetime is reset upon each successful login in which the **Remember my username** checkbox on the login form is selected. ⓘ **Note** The value is ignored when users authenticate through a Composite Adapter instance that chains this adapter behind another authentication source with an **Input User ID Mapping** configuration, and the **Allow Username Edits** checkbox is not selected. You can enter an integer between `1` and `3650`. The default value is `30` days.
'This is My Device' Lifetime	The number of days that a user's selection of the **This is my device** checkbox on the login form is retained. The lifetime is reset upon each successful login in which the **This is my device** checkbox on the login form is selected. You can enter an integer between `1` and `3650`. The default value is `30` days.
Allow Username Edits During Chaining	When users authenticate through a Composite Adapter instance that chains this adapter behind another authentication source with an **Input User ID Mapping** configuration or initiate an OAuth authorization request with a `login_hint` parameter, the username in the login form is pre-populated. Users are not allowed to edit their usernames. Select this checkbox if you want to allow users to edit the pre-populated username in the login form. ⓘ **Note** Users who authenticate through a Composite Adapter instance without an **Input User ID Mapping** configuration or this adapter directly always need to enter their usernames. This checkbox is not selected by default.

Property	Description
Track Authentication Time	When selected, the time of authentication for each user is tracked and can be utilized by applicable use cases. For example, if an OAuth client sends an authorization request with a `max_age` parameter, the request prompts the user to reauthenticate when the elapsed time between the current time and the time of the previous authentication is greater than the `max_age` value. This checkbox is selected by default.
Post-Password Change Re-Authentication Delay	The HTML Form Adapter reauthenticates the user using the new password immediately after a successful password change request. As needed, enter the amount of time, in milliseconds, that the adapter can wait prior to the reauthentication attempt. The default value is `0`, which is the minimum value. The maximum value is `60000`, or 1 minute.

Advanced fields for self-service password reset and account unlock

Property	Description
Password Reset One-Time Link Email Template	The HTML template to send the user an email with a password reset link when **Password Reset Type** is **Email One-Time Link**.
Password Reset One-Time Password Email Template	The HTML template to send the user an email with a one-time password reset code when **Password Reset Type** is **Email One-Time Password**. The default template file is `message-template-forgot-password-code.html`.
Password Reset Complete Email Template	The HTML template to send the user an email that the password reset is complete. The default template file is `message-template-forgot-password-complete.html`.
Password Reset Failed Email Template	The HTML template to send the user an email that the password reset attempt failed. The default template file is `message-template-forgot-password-failed.html`.
Password Reset Code Template	The HTML template to prompt the user to enter the one-time passcode (OTP) for password reset. This template applies when the password reset type is **Email One-Time Password** or **Text Message**. The default template file is `forgot-password-resume.html`.
Password Reset Template	The HTML template to prompt the user to define a new password. This template applies for all password reset types other than **None**. The default template file is `forgot-password-change.html`.
Password Reset Error Template	The HTML template to notify the user that the password reset attempt has failed. This template applies for all password reset types other than **None**. The default template file is `forgot-password-error.html`.

Property	Description
Password Reset Success Template	The HTML template to notify the user that the password reset attempt has succeeded. This template applies for all password reset types other than **None**. The default template file is `forgot-password-success.html`.
Account Unlock Template	The HTML template to notify the user that the account unlock attempt has succeeded and to prompt the user to retain the current password or reset it. The default template file is `account-unlock.html`.
Account Unlock Email Template	The HTML template to send the user an email that the account unlock attempt has succeeded. The default template file is `message-template-account-unlock-complete.html`.
OTP Length	The number of characters in the one-time password for password reset. The default value is `8`.
Allowed OTP Character Set	The alphanumeric characters that PingFederate can include in an OTP. The default value is `23456789BCDFGHJKMNPQRSTVWXZbcdfghjkmnpqrstvwxz`. **Note** You must enter a minimum of 10 characters. Provide unique characters to ensure a secure OTP.
Password Reset Token Validity Time	The validity in minutes for the OTP or the one-time link. The default value is `10` minutes.
PingID Properties	To configure self-service password reset using PingID, you must obtain the `PingID.properties` file and upload it to the HTML Form Adapter instance. 1. Sign on to the PingOne admin portal. 2. Go to **Setup** > **PingID** > **Client Integration**. 3. Download the settings file `PingID.properties`. 4. Close the PingOne admin portal. 5. On the **Manage IdP Adapters** tab in the PingFederate administrative console, click **Choose File**. 6. Select the `PingID.properties` file and click **Open**.

Note

When configuring an adapter to use a custom template name, make sure the `pingfederate/server/default/conf/language-packs/pingfederate-email-messages.properties` file and any language specific version, such as `pingfederate-email-messages_fr.properties`, includes that name so that the email subject found in the properties file is used.

For example, to customize an adapter to use a new password reset complete email template using `my-template-forgot-password-complete.html`, add the new property with the email's subject text. The new entry should be `my-template-forgot-password-complete.html=Password Reset`.

Find the configurable text that applies to a specific template in the `pingfederate-email-messages.properties` file, and make sure the same key-value pairs are specified for their new template name.

Advanced fields for self-service username recovery

Property	Description
Require Verified Email	When selected, PingFederate requires that the user's email address is verified before sending a password reset, account unlock, or username recovery email. If users are permitted to manage their accounts, they will be blocked from accessing any connected application until they have verified their email. Learn more about enabling user account management in Configuring local identity profiles. By default, the checkbox is cleared.
Username Recovery Template	The HTML template to prompt the user to enter an email address to recover the username associated with the account. This template applies when username recovery is enabled. The default template file is `username.recovery.template.html`.
Username Recovery Info Template	The HTML template to notify the user to retrieve the email message with the recovered username. This template applies when username recovery is enabled. The default template file is `username.recovery.info.template.html`.
Username Recovery Email Template	The HTML email template PingFederate uses to generate the email message that contains the recovered username. The default template file is `message-template-username-recovery.html`, located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory. Applicable only if an instance of the SMTP Notification Publisher is selected in the **Notification Publisher** list.

Risk options

Property	Description
Risk Provider	Select a risk provider. The default selection is **Default**, which is the risk provider specified as the default on the **CAPTCHA and Risk Providers** page. To add a risk provider instance, click **Manage CAPTCHA and Risk Providers** to open the **CAPTCHA and Risk Providers** page, then follow the steps in Managing CAPTCHA and risk providers.
Risk for Authentication	Enables risk to protect the authentication process from automated attacks.
Risk for Password Change	Enables risk to protect the password change process from automated attacks.
Risk for Password Reset	Enables risk to protect the account recovery process for password reset and account unlock from automated attacks.
Risk for Username Recovery	Enables risk to protect the username recovery process from automated attacks.

By default, risk checkboxes are cleared.

Other settings

Property	Description
Fail Authentication on Account Lockout	This setting determines the adapter's behavior when PingFederate locks a user's account due to too many failed login attempts. • When selected, the adapter returns an error and policy processing continues on the FAIL branch. • When cleared, the adapter displays the login page with an error message stating the account is locked. The checkbox is cleared by default.

Variables available to HTML Form Adapter templates

The following variables are available to the HTML Form Adapter templates for core templates as well as password reset, change password, and username recovery use cases:

- `$adapterId` - The IdP adapter ID used in this transaction

- `$baseUrl` - The base URL of the PingFederate instance
- `$client_id` - The ID of the OAuth client used in this transaction
- `$connectionName` - The name of the SP connection used in this SSO transaction
- `$entityId` - The entity ID (connection ID) of the SP connection used in this SSO transaction
- `$spAdapterId` - The SP adapter ID used in this transaction
- `$userAttributes` - The user-specific data retrieved from the template type used in this transaction. The `$userAttributes` variable represents the attributes associated with a user's identity and enables the retrieval of user-specific information across templates

> **Note**
>
> Variables are populated when applicable.
> The core templates are:
>
> - `html.form.login.template.html`
> - `html.form.message.template.html`
> - `html.form.password.expiring.notification.template.html`
> - `html.form.login.challenge.template.html`

Related links

- Invisible reCAPTCHA documentation
- Managing CAPTCHA and risk providers
- Configuring the LDAP Username Password Credential Validator
- Configuring self-service account recovery
- Configuring self-service user name recovery
- Configuring authentication sessions
- Customizable user-facing pages
- Customizable email notifications
- Customizable text message
- Localizing messages for end users

HTTP Basic Adapter

The HTTP Basic adapter provides user authentication through a password credential validator (PCV) to integrate PingFederate with local authentication mechanisms.

Initial user authentication is normally handled outside of the PingFederate server using an application or an Identity Management system authentication module. Adapters or agents from PingFederate integration kits are typically used to integrate with these local authentication mechanisms.

PingFederate packages an HTTP Basic Adapter that delegates user authentication to a Password Credential Validator, such as an LDAP Username PCV (see Password Credential Validators). This authentication mechanism validates credentials against a user repository through an instance of a PCV. You can add multiple PCV instances to an instance of the HTTP Basic Adapter to validate against multiple user repositories, in which case PingFederate falls to the subsequent PCV instance if the previous PCV instance fails to validate the user credentials.

When PingFederate receives an authentication request and the use case is associated with an HTTP Basic Adapter instance, PingFederate invokes the adapter if it does not find a valid authentication session (see Sessions). If the HTTP Basic Adapter does not find a valid adapter session, it prompts the user for credentials.

This adapter does not provide an authentication context. For SAML connections, PingFederate sets the authentication context as follows:

- `urn:oasis:names:tc:SAML:1.0:am:unspecified` for SAML 1.x
- `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` for SAML 2.0

PingFederate can override the authentication context with either an instance of the Requested AuthN Context Authentication Selector or the `SAML_AUTHN_CTX` attribute in the SAML attribute contract. The latter takes precedence.

Configuring an HTTP Basic Adapter instance

Configure an HTTP Basic Adapter instance to use credentials against a user repository through an instance of a password credential validator (PCV) to support user authentication when it occurs outside of the PingFederate server.

About this task

Using the administrative console, configure an HTTP Basic Adapter instance.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.
    3. Optional: In the **Parent Instance** list, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. On the **IdP Adapter** tab, configure your HTTP Basic Adapter instance as follows:
    1. If you have not yet defined the desired Password Credential Validator instance, click **Manage Password Credential Validators** to do so.
    2. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.

3. From the **Password Credential Validator Instance** list, select a Password Credential Validator instance. Click **Update**.Add as many validators as necessary. Use the up and down arrows to adjust the order in which you want PingFederate to attempt credential authentication. If the first mechanism fails to validate the credentials, PingFederate moves sequentially through the list until credential validation succeeds. If none of the Password Credential Validator instances can authenticate the user's credentials, and the challenge retries maximum has been reached, the process fails.

   > **Note**
   >
   > If usernames overlap across multiple Password Credential Validator instances, this failover setup could lock out those accounts in their source locations.

4. Enter values for the adapter configuration.

   See the on-window field descriptions and the following table for more information.

PingFederate's fields and descriptions for creating an HTTP Basic Adapter instance

Property	Description
Realm (Required)	The name of a protected area. The value of this field is sent as a part of the HTTP Basic authentication request. It appears in a dialog box that prompts the user for a username and password. + **Note** After a user authenticates against a realm, if additional HTTP Basic Adapter instances share the same realm, the user is not prompted to re-authenticate.
Challenge Retries (Required)	The number of attempts allowed for password authentication. The default value is `3`.

2. On the **Extended Contract** window, configure additional attributes for this adapter instance as needed.

   The HTTP Basic Adapter contract includes one core attribute: `username`.

3. On the **Adapter Attributes** tab, do the following:

   1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

      The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

      > **Note**
      >
      > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

> **Important**
>
> For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

   This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

   > **Note**
   >
   > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

   > **Note**
   >
   > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

4. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.
   - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
   - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

5. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

6. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

   If you want to exit without saving the configuration, click **Cancel**.

Identifier First Adapter

The Identifier First Adapter works best for use cases when a variety of user types are authenticating with PingFederate. The adapter analyzes the type of user and the credentials with which they have enrolled before, including datastore queries and user attributes, to provide support for user authentication.

When PingFederate receives an authentication request and the use case is associated with an Identifier First Adapter instance, PingFederate invokes the adapter if it does not find a valid authentication session. The adapter prompts the user to enter their identifier and captures the identifier in the `subject` attribute.

> **Note**
>
> `subject` is one of the two core attributes in the adapter contract. `domain` is the other one.

If the identifier is an email address, the adapter extracts the email address suffix and exposes it downstream through the `domain` attribute. Additionally, the adapter can leverage datastore queries to fulfill the `domain` attribute, or other extended attributes, to support identifiers of other kinds.

Based on the identification result and the configured authentication policies, PingFederate routes the user to the desired policy path. As the user fulfills the authentication requirements, the adapter preserves the identifier on the client side in a persistent cookie.

When the user signs off and makes a subsequent sign-on request from the same browser, the adapter offers the user to either select the previously authenticated identifier found in the cookie or to enter a new one. If the user opts to enter a new identifier, the adapter adds that identifier to the cookie after the user completes the authentication requirements.

The adapter keeps adding the most-recently-authenticated identifier until the number of identifier reaches a configurable limit. When the threshold is reached, the adapter removes the least-recently-used identifier from the cookie.

Lastly, the Identifier First Adapter also allow users to continue without entering or selecting an identifier, in which case it treats the authentication attempt as a failure and returns control to PingFederate. PingFederate can then route the request based on the configured policy path.

> **Tip**
>
> PingFederate creates authentication sessions configured for an Identifier First Adapter instance only when the complete single sign-on (SSO) transaction has succeeded. This lets the adapter prompt the user for a different user identifier when a chained adapter authentication fails because, for example, there's a typo in the user identifier.

> **Note**
>
> The Identifier First Adapter is authentication API-capable. For more information, see Authentication applications and the authentication API.

Configuring an Identifier First Adapter instance

Configure an instance of the Identifier First Adapter in PingFederate following these instructions and for additional configuration information on fieldnames, descriptions, and optimal settings depending on your use case.

About this task

Using the PingFederate administrative console, configure an Identifier First Adapter instance.

Steps

1. Go to **Authentication > Integration > IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.
    3. Optional: In the **Parent Instance** list, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. On the **IdP Adapter** tab, configure your Identifier First Adapter instance.

   For more information about each field, see the following table.

PingFederate's fields and descriptions for creating an Identifier First Adapter instance

Field	Description
Identifier Cookie Lifetime	Determines the number of days that previously authenticated identifiers are preserved as a cookie on the client side. This value can range from `0` through `3650`. Set to `0` to disable the storage of any previously authenticated identifiers. The default value is `30`.
Allow Cancelling Identifier Selection	Determines whether a user is allowed to continue without entering or selecting an identifier. If allowed, when a user decides to continue without providing an identifier, the Identifier First Adapter treats the authentication attempt as a failure and returns control to PingFederate. This check box is not selected by default.
Click **Show Advanced Fields** to review the following settings. Modify as needed.	
Maximum Identifiers Count	Determines the maximum number of previously authenticated identifiers can be preserved in the identifier cookie. This value can range from `0` through `10`. Set to `0` to disable the storage of any previously authenticated identifiers. The default value is `5`.
Identifier Selection Template	The HTML template to prompt the user to enter or select an identifier. PingFederate allows each configured adapter instance to use a different template as needed. The default template file is `identifier.first.template.html`. Like other Velocity template files, it is located in the `<pf_install>/pingfederate/server/default/conf/template` directory.

2. On the **Extended Contract** tab, configure additional attributes for this adapter instance as needed.

The Identifier First Adapter contract includes two core attributes: `subject` and `domain`.

If the identifier is an email address, the adapter extracts the email address suffix and exposes it downstream through the `domain` attribute. As needed, the adapter can leverage datastore queries to fulfill the `domain` attribute. For more information, see step 7).

3. On the **Adapter Attributes** tab, do the following:

    1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

        The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

        > **Note**
        >
        > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

        > **Important**
        >
        > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

    2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

        This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

        > **Note**
        >
        > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

    3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

        > **Note**
        >
        > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

    4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

4. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

    ◦ Configure one or more data sources for datastore queries.

- Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
- Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

5. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

6. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

   If you want to exit without saving the configuration, click **Cancel**.

Related links

- Customizable user-facing pages

Identifier First Adapter and authentication policies

The Identifier First Adapter works best in conjunction with authentication policies and setting expected attribute values to enforce authentication requirements.

The Identifier First Adapter is designed to identify user populations. It supports email addresses natively: it extracts the email address suffix and exposes it downstream through the `domain` attribute. Additionally, the adapter can leverage datastore queries to fulfill the `domain` attribute or other extended attributes to support identifiers of other kinds.

The Identifier First Adapter is most effective when used in conjunction with authentication policies. The policy paths are created by having rules matching expected values of the `domain` attribute or other extended attribute. Each expected value forms its own policy path, to which a series of authentication sources can be appended to enforce the desired authentication requirements.

For more information and configuration steps, see the subsequent sample use case.

Configuring a policy for multiple user populations

Configure an Identifier First Adapter instance to determine user populations based on user identifiers usernames and an authentication policy to route sign-on requests to authentication sources tailored for their respective user populations.

About this task

Using the administrative console, follow the instructions below for configuring an Identifier First Adapter instance, creating an authentication policy to prompt the user for their identifier first, determining their user population, and routing the request to the desired authentication recommendations. Consider the sample use case here.

You need to enforce different sets of authentication requirements for two sets of users, employees, and external consultants.

Employees are given username@example.com email addresses, such as asmith@example.com. User records are stored in a local directory server. Employees sign on through an HTML Form Adapter instance.

Consultants have either username@example.org or username@example.info email addresses. User records are stored in a local database. Consultants can sign on using their username or email address and password through a local web portal. This web portal is integrated with PingFederate using the OpenToken framework.

Your organization owns another local database that keeps track of username, domain information, and email address for both employees and consultants. The column names are `dsUid`, `dsDomain`, and `dsMail`, respectively. For simplicity, no users share the same `dsUid` value.

In this sample use case, you must make sure that the Identifier First Adapter instance can handle the scenario where users might enter their email address or just their username when setting up the Identifier First Adapter instance. Additionally, when accessing protected resources, your organization has agreed to send the user's email address in the security token.

You have already created the following components:

- An LDAP datastore connecting to the local directory server. The attribute name of the user identifier is `uid`.
- An instance of the LDAP Username Password Credential Validator (PCV) validating credentials against the local directory server with the LDAP datastore. The LDAP Username PCV instance is extended with an additional attribute `mail`. The search filter is configured to handle identifiers in the format of an email address or a username. See the following code example. + `(\|(uid=${username})(mail=${username}))`
- An HTML Form Adapter instance delegating credential-validation to the LDAP Username PCV instance. The HTML Form Adapter instance is also extended with an additional attribute `mail`, which takes the `mail` attribute value from the LDAP Username PCV instance. The ID of this HTML Form Adapter instance is `htmlForm`.
- An OpenToken IdP Adapter instance digesting tokens from the web portal as the source of user attributes. The adapter contract is extended with an additional attribute `mail`. The web portal is designed to always include the user's email address in the token through the `mail` attribute. The ID of this OpenToken IdP Adapter instance is `opentTokenIdp`.

This sample use case requires the following additional components:

- An expression-enabled PingFederate environment. See step 1.
- An authentication policy contract to carry the email address from your organization to your partners. See step 2.
- A Java Database Connectivity (JDBC) datastore connecting to the database that hosts username, email, and domain information. See step 3.
- An Identifier First Adapter instance with an attribute source lookup configuration and a contract fulfillment via expressions for the `domain` adapter attribute. See step 4.
- An authentication policy to route user requests to different authentication sources based on user populations. See step 5.

To fulfill the requirements:

Steps

1. Enable expressions in PingFederate.

   For configuration steps, see Enabling and disabling expressions.

2. Go to **Authentication** > **Policies** > **Policy Contracts**.
3. On the **Policy Contracts** window, click **Create New Contract** and create an authentication policy contract without any additional attributes.
4. Go to **System** > **Data & Credential Stores** > **Data Stores**.
5.

From the **Data Stores** window, click **Add New Data Store**.

1. On the **Data Store Type** tab, enter **Name**.
2. From the **Type** list, select **Database (JDBC)** to create a JDBC datastore connection to the database that hosts username and domain information.

6. Create an instance of the Identifier First Adapter instance.

   1. Follow steps 1 through 6 in Configuring an Identifier First Adapter instance.

      For the sample use case, suppose you name the adapter instance `ID 1st`.

   2. Go to **Applications > Integration > Adapter-to-Adapter Mappings**.
   3. On the **Adapter-to-Mappings** window, select a source instance and a target instance, and click **Add Mapping**.
   4. On the **Attribute Sources & User Lookup** tab, click **Add Attribute Source**.

      > **Note**
      >
      > For more information about configuring the following steps, see Datastore query configuration.

      1. On the **Data Store** tab, enter an ID in **Attribute Source ID** and a name in **Attribute Source Description**, such as `domainInfo` and `Domain Info`, respectively. In the **Active Data Store**list, select the JDBC datastore created previously. Click **Next**.
      2. On the **Database Table and Columns** tab, select the applicable options from the **Schema** and **Table** lists.
      3. Under **Columns to return from SELECT**, select the `dsDomain` column and click **Add Attribute**. Click **Next**.
      4. On the **Database Filter** tab, in the **Where** field, specify a filter to search by identifier that can handle identifiers in the format of an email address or a username.

         See the following example of a filter entry. `dsUid='${subject}' OR dsMail='${subject}'`.

      5. Click **Next**.
      6. On the **Summary** tab, click **Done**.

   5. On the **Adapter Contract Fulfillment** tab, configure as follows.

Contract	Source	Value
domain	Expression	+ `#this.get("domain").toString().matches("(?i).+") ? #this.get("domain") : #this.get("ds.domainInfo.dsDomain")` **Note** Line breaks are inserted for readability only.

Contract	Source	Value
subject	Adapter	Not applicable. No selection is required.

The expression checks the `domain` attribute value returned by the Identifier First Adapter. If the value contains one or more character, PingFederate uses that as the value for the `domain` attribute. Otherwise, it uses the `dsDomain` column value returned from the JDBC datastore. In other words, this expression handles identifiers in the format of an email address or a username.

This sample expression is intended to demonstrate the capability of the Identifier First Adapter. Depending on the actual use cases, expressions may vary. For more information about expressions, see Construct OGNL expressions.

6. On the **Issuance Criteria** tab, click **Next**.

> **Note**
>
> Depending on the actual use cases, you can add issuance criteria.

7. On the **Adapter-to-Adapter Summary** tab, review your configuration instance. Click **Done** to save your adapter instance configuration.

7. Create an authentication policy with rules to form policy paths based on results from `domain` attribute values returned by the Identifier First Adapter.

   1. Go to **Authentication** > **Policies** > **Policies**.

   2. From the **Policies** tab, click **Add Policy**.

   3. On the **Policy** window, enter a **Name**, and optionally a **Description**, for the policy.

   4. From the **Policy** list, select the Identifier First Adapter instance created in step 5.

   5. Click **Rules** to open the **Rules** dialog.

   6. Add three rules as follows.

Defining Authentication Policy Rules dialog fields and entries

Attribute Name	Condition	Value	Result
domain	equal to	example.com	Example COM
domain	equal to	example.org	Example ORG
domain	equal to	example.info	Example INFO

> **Note**
>
> Add one rule for each expected `domain` attribute value.

7. Clear the **Default to Success** check box to disable the option to specify a policy path for the scenario where the `domain` attribute value from the Identifier First Adapter instance does not match any configured value on the **Rules** dialog.

   If you want to enable an authentication policy path for unexpected `domain` attribute values, leave the **Default to Success** check box as selected.

   For more information about rules, see Configuring rules in authentication policies.

8. Click **Done** to close the **Rules** dialog.

   Result:

   By adding three rules and disabling the default to success option, the Identifier First Adapter instance now contains four policy paths: Fail, Example COM, Example ORG, and Example INFO.

9. Configure each policy path.

   Fail

   Select **Done**, which terminates the request in an error condition.

   Example COM

   Select the HTML Form Adapter instance, which contains two paths: Fail and Success.Configure each policy path.

   Fail

   Select **Done**, which terminates the request in an error condition.

   Success

   Select the policy contract created in step 2.

   Click **Options** to open the **Incoming User ID** dialog.

   1. From the **Source** list, select **Adapter (ID 1st)**.
   2. From the **Attribute** list, select **subject**.
   3. Click **Done** to close the **Incoming User ID** dialog.

      For more information, see Specifying incoming user IDs.

   Example ORG (and then Example INFO)

   Select the OpenToken IdP Adapter instance, which contains two paths: Fail and Success.Configure each policy path by using the same steps documented for the **Example COM** policy path

10. Configure contract fulfillment for each authentication policy contract as follows.

Contract Fulfillment fieldnames and entries

Result from rules	Contract Attribute	Source	Value
Example COM	subject	Adapter (htmlForm)	mail
Example ORG	subject	Adapter (openTokenIdp)	mail
Example INFO	subject	Adapter (openTokenIdp)	mail

For more information, see Configuring contract mapping.

11. Click **Done**. Click **Save**.

Result

You have now successfully configured an Identifier First Adapter instance and an authentication policy to prompt the user for their identifier first, determine their user population, and route the request to the desired authentication policy path.

Related links

- Managing policy contracts
- Configuring a JDBC connection

Kerberos Adapter

The integrated Kerberos Adapter provides a seamless single sign-on (SSO) experience for Windows clients by authenticating SSO requests using the Kerberos v5 protocol against Active Directory (AD) domains.

When the PingFederate Identity Provider (IdP) server receives an authentication request for Service Provider-initiated SSO or a user clicks a hyperlink for IdP-initiated SSO, PingFederate invokes the Kerberos Adapter and returns to the browser an HTTP 401 Unauthorized response. When PingFederate receives a Kerberos ticket from the browser, it validates the ticket against the domain defined in the Kerberos Adapter configuration. If validation succeeds, PingFederate retrieves the username, the domain, and the security identifiers (ObjectSID and SIDs) from the ticket; generates a SAML assertion with the username and optionally the associated domain, security identifiers, or both; and passes it to the SP.

> **Note**
>
> The Kerberos Adapter supports authentications by Kerberos only.

Authentication mechanism assurance

The integrated Kerberos Adapter supports authentication mechanism assurance from Active Directory domain service.

With an Identity Provider (IdP), you can use the Token Authorization framework to verify the `ObjectSID` and `SIDs` values before issuing a token. Alternatively, you can map the `SIDs` value to an attribute in the contract and let the Service Provider (SP) determine if the user meets the requirements to access the protected resource. For the purpose of protecting resources based on sign-on method, authentication mechanism assurance from Active Directory (AD) domain service adds an additional group membership to the user's security identifiers attribute `SIDs` when a user signs on using a certificate-based sign-on method, such as a smart-card sign-on For example, you can restrict access to sensitive resources to users who sign on by using their smart cards, which requires a physical reader that you place in a physically secured location.

The integrated Kerberos Adapter supports authentication mechanism assurance by including the `ObjectSID` and `SIDs` attributes of the authenticated user in the adapter contract.

If your use case requires authentication mechanism assurance, you can add a criterion in the Token Authorization framework to verify that the `SIDs` attribute contains the security identifier (SID) value associated with the required login method. If the `SIDs` attribute does not contain the specified SID value, the request is denied.

> **Note**
>
> The `SIDs` attribute contains multiple values. Use the **multi-value contains** condition or the **multi-value contains (case insensitive)** condition to verify whether the `SIDs` attribute contains a specific value. You can also configure more complex evaluations using OGNL expressions.

Alternatively, you can map the `ObjectSID` and `SIDs` attributes into the contract and let the SP determine if the user meets the requirements to access the protected resource.

For more information about authentication mechanism assurance, see the Authentication Mechanism Assurance for AD DS in Windows Server 2008 R2 Step-by-Step Guide from Microsoft's documentation.

Configuring a Kerberos Adapter instance for SSO authentication

When integrating PingFederate with Windows client applications so that they can use single sign-on to authenticate, create and configure an instance of the Kerberos Adapter.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
   1. Enter the **Instance Name** and **Instance ID**.
   2. In the **Type** list, select the adapter type.
   3. Optional: In the **Parent Instance** list, select an existing type.

      If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. On the **IdP Adapter** window, configure your Kerberos Adapter instance.

   See the on-window field descriptions and the following table for more information.

Field	Description
Domain/Realm Name (Required)	Select your Windows domain. If the domain or realm you want does not appear, click **Manage Active Directory Domains/Kerberos Realms** to add it. For more information, see Active Directory and Kerberos.
Error URL Redirect	Enter a URL for redirecting the user if there are errors. This URL has an `errorMessage` query parameter appended to it, which contains a brief description of the error that occurred. The error page can optionally display this message on the window to provide guidance on remedying the problem. ⓘ **Note** In the case of an error, if you define an **Error URL Redirect** and the adapter instance is included in an instance of the Composite Adapter, the user is redirected to the configured error URL rather than continuing on to the next adapter in the chain. Leave this field blank to have the adapter continue on to the next adapter. When employing the `errorMessage` query parameter in a custom error page, adhere to Web-application security best practices to guard against common content injection vulnerabilities. If no URL is specified, the appropriate default error landing page appears.
Click **Show Advanced Fields** to review the following settings. Modify as needed.	
Error Template	When selected, displays a template to provide standardized information to the end user when authentication fails. The **Error URL Redirect** value is ignored. The template `kerberos.error.template.html` in the `<pf_install>/pingfederate/server/default/conf/template` directory uses the Velocity template engine and can be modified in a text editor to suit your particular branding and informational needs. For example, you can give the user the option to try again if authentication fails. For more information on Velocity templates, see Customizable user-facing pages.
Fail when Re-authentication is Requested	When the check box is selected, if PingFederate receives an authentication request containing a re-authentication parameter, the adapter will respond with a failure status so that the policy's failure branch is followed. For OAuth 2.0, the re-authentication parameter is `ForceAuthn=true`. For OpenID Connect, the re-authentication parameter is `prompt=login`. By default, the check box is cleared.

Field	Description
Authentication Context Value	This can be any value agreed to with your SP partner to indicate the type of credentials used to authenticate. Standard URIs are defined in the SAML specifications. For more information on SAML specifications, see the OASIS documents and saml-authn-context-2.0-os.pdf. If left blank, PingFederate sets the authentication context as follows: ◦ `urn:oasis:names:tc:SAML:1.0:am:unspecified` for SAML 1.x ◦ `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` for SAML 2.0 Either an instance of the Requested AuthN Context Authentication Selector or the `SAML_AUTHN_CTX` attribute can override the authentication context in the SAML attribute contract. The latter takes precedence.

2. On the **Extended Contract** tab, configure additional attributes for this adapter instance as needed.

   The Kerberos Adapter contract includes four core attributes: `Domain/Realm Name`, `ObjectSID`, `SIDs`, and `Username`.

3. On the **Adapter Attributes** tab, do the following:

   1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

      The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

      > **Note**
      >
      > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

      > **Important**
      >
      > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

   2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

      This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

      > **Note**
      >
      > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

   3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

> **Note**
>
> Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

4. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.
   - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
   - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

5. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

6. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

   If you want to exit without saving the configuration, click **Cancel**.

Related links

- Customizable user-facing pages

Configuring browsers for Kerberos authentication

You can configure browsers at your site to use the Kerberos Adapter to authenticate users.

The client-side configuration requires the base URL or an applicable virtual host name of your PingFederate environment. The base URL is defined on the **System > Server > Protocol Settings > Federation Info** tab. To see a list of defined virtual host names, if configured, go to **System > Server > Virtual Host Names**.

The following information explains how to configure the Microsoft Edge, Mozilla Firefox, and Google Chrome browsers.

> **Important**
>
> If the browser is not properly configured, the user might be prompted to authenticate manually with their network credentials. Otherwise, authentication fails the single sign-on (SSO) to the service providers.

Microsoft Edge

Configuring Microsoft Edge for Kerberos authentication

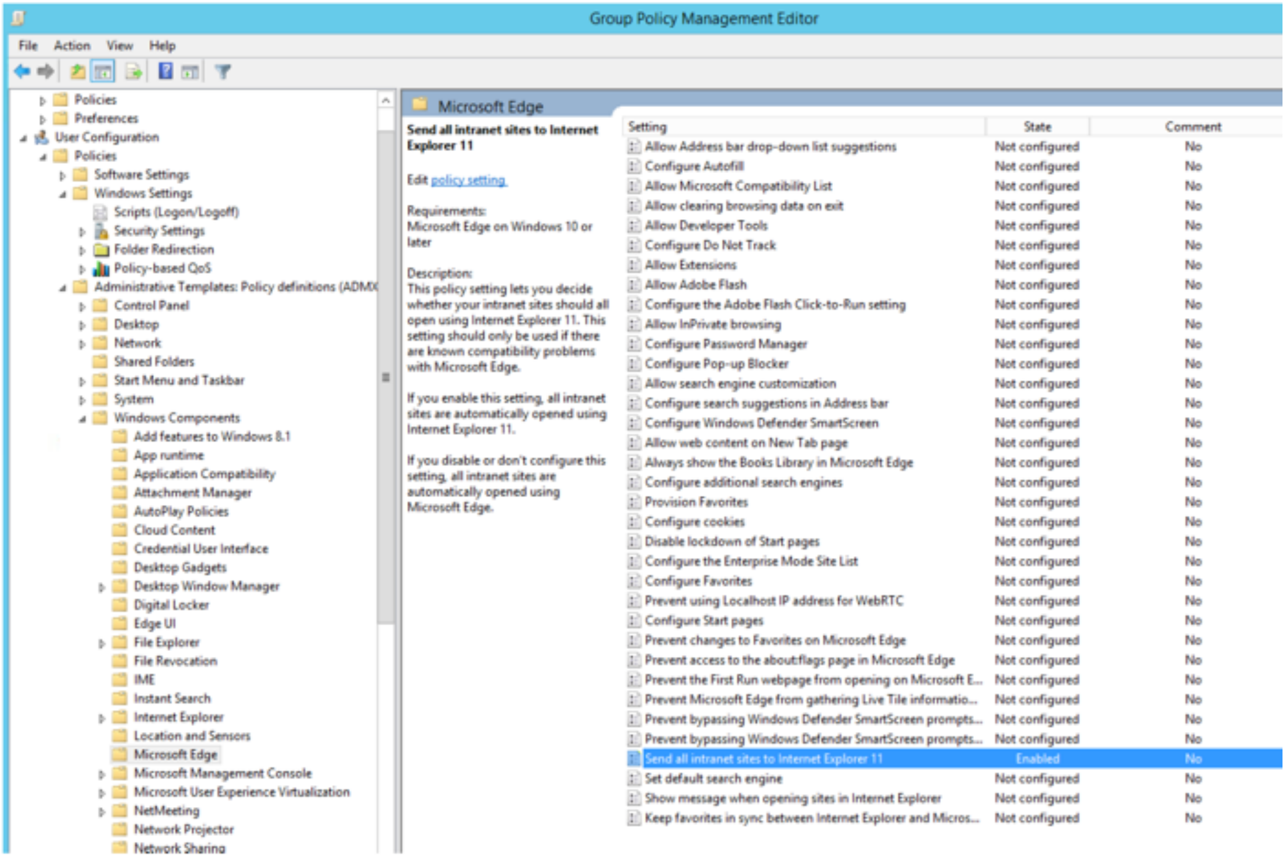
You can configure Microsoft Edge browsers for Kerberos authentication.

About this task

Because Edge doesn't honor intranet sites, the PingFederate Kerberos Adapter isn't allowed by default to request the Kerberos ticket for a user. To resolve this issue, there's a group policy object (GPO) that can send intranet site requests to Internet Explorer 11 instead of Edge. It lets you put PingFederate in the Intranet Sites Zone (not the Trusted Sites Zone) in Internet Explorer and enable Kerberos.

Steps

1. In the Group Policy Management editor, go to **User Configuration > Administrative Templates > Windows Components > Microsoft Edge** and enable the **Send All intranet sites to IE11** setting.



Send all intranet sites to Internet Explorer 11

Edit policy setting

Requirements:
Microsoft Edge on Windows 10 or later

Description:
This policy setting lets you decide whether your intranet sites should all open using Internet Explorer 11. This setting should only be used if there are known compatibility problems with Microsoft Edge.

If you enable this setting, all intranet sites are automatically opened using Internet Explorer 11.

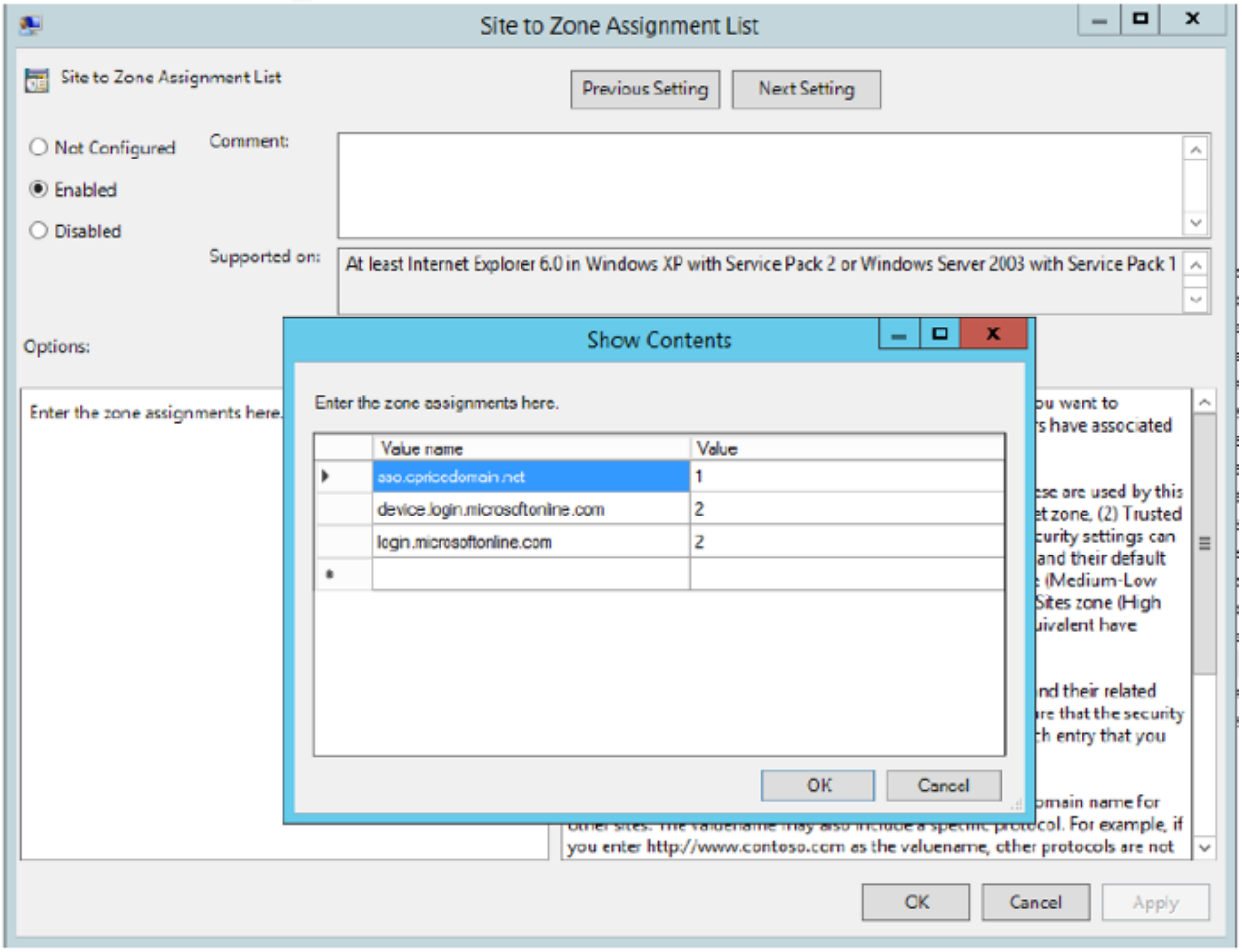
If you disable or don't configure this setting, all intranet sites are automatically opened using Microsoft Edge.

Setting	State	Comment
Allow Address bar drop-down list suggestions	Not configured	No
Configure Autofill	Not configured	No
Allow Microsoft Compatibility List	Not configured	No
Allow clearing browsing data on exit	Not configured	No
Allow Developer Tools	Not configured	No
Configure Do Not Track	Not configured	No
Allow Extensions	Not configured	No
Allow Adobe Flash	Not configured	No
Configure the Adobe Flash Click-to-Run setting	Not configured	No
Allow InPrivate browsing	Not configured	No
Configure Password Manager	Not configured	No
Configure Pop-up Blocker	Not configured	No
Allow search engine customization	Not configured	No
Configure search suggestions in Address bar	Not configured	No
Configure Windows Defender SmartScreen	Not configured	No
Allow web content on New Tab page	Not configured	No
Always show the Books Library in Microsoft Edge	Not configured	No
Configure additional search engines	Not configured	No
Provision Favorites	Not configured	No
Configure cookies	Not configured	No
Disable lockdown of Start pages	Not configured	No
Configure the Enterprise Mode Site List	Not configured	No
Configure Favorites	Not configured	No
Prevent using Localhost IP address for WebRTC	Not configured	No
Configure Start pages	Not configured	No
Prevent changes to Favorites on Microsoft Edge	Not configured	No
Prevent access to the about:flags page in Microsoft Edge	Not configured	No
Prevent the First Run webpage from opening on Microsoft E...	Not configured	No
Prevent Microsoft Edge from gathering Live Tile informatio...	Not configured	No
Prevent bypassing Windows Defender SmartScreen prompts...	Not configured	No
Prevent bypassing Windows Defender SmartScreen prompts...	Not configured	No
Send all intranet sites to Internet Explorer 11	Enabled	No
Set default search engine	Not configured	No
Show message when opening sites in Internet Explorer	Not configured	No
Keep favorites in sync between Internet Explorer and Micros...	Not configured	No

2. Go to **Administrative Templates > Windows Components > Internet Explorer > Internet Control Panel > Security Page > Site to Zone Assignment List**.

3. In the **Show Contents** dialog box's **Value Name** column enter the *<PingFederate URL>*.

4. In the **Value** column enter `1`.



○ Not Configured

● Enabled

○ Disabled

Supported on: At least Internet Explorer 6.0 in Windows XP with Service Pack 2 or Windows Server 2003 with Service Pack 1

Enter the zone assignments here.

Value name	Value
sso.cpricedomain.net	1
device.login.microsoftonline.com	2
login.microsoftonline.com	2

ou want to
s have associated

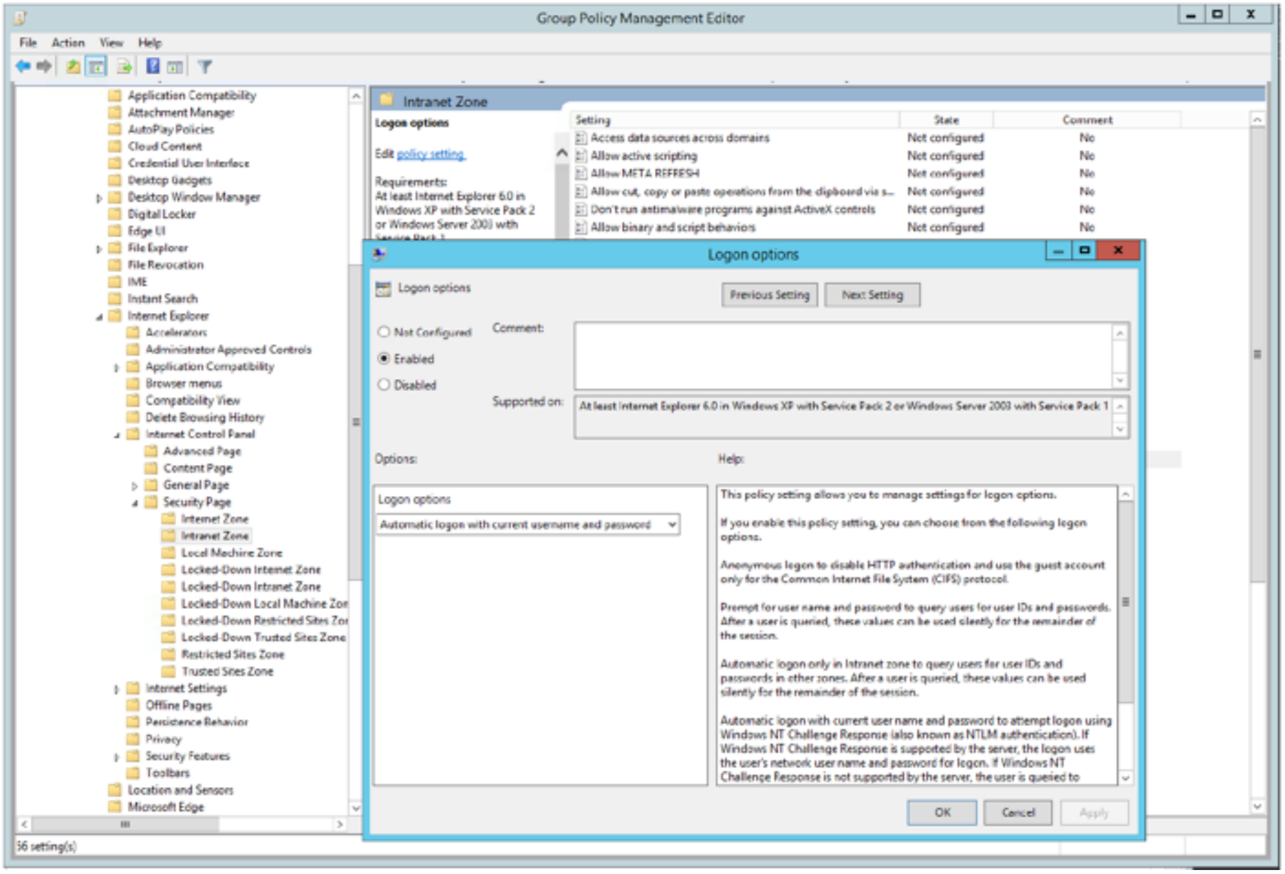
ese are used by this
et zone, (2) Trusted
curity settings can
and their default
: (Medium-Low
Sites zone (High
uivalent have

nd their related
re that the security
h entry that you

omain name for
other sites. The valuename may also include a specific protocol. For example, if
you enter http://www.contoso.com as the valuename, other protocols are not

5. Go to **User Configuration** > **Administrative Templates** > **Windows Components** > **Internet Explorer** > **Internet Control Panel** > **SecurityPage** > **Intranet Zone**.

6. In the **Logon Option** dialog box's **Logon options** list, select **Automatic logon with current username and password**.



Logon options

Edit policy setting.

Requirements:
At least Internet Explorer 6.0 in Windows XP with Service Pack 2 or Windows Server 2003 with Service Pack 1

Setting	State	Comment
Access data sources across domains	Not configured	No
Allow active scripting	Not configured	No
Allow META REFRESH	Not configured	No
Allow cut, copy or paste operations from the clipboard via s...	Not configured	No
Don't run antimalware programs against ActiveX controls	Not configured	No
Allow binary and script behaviors	Not configured	No

Logon options

○ Not Configured
● Enabled
○ Disabled

Comment:

Supported on: At least Internet Explorer 6.0 in Windows XP with Service Pack 2 or Windows Server 2003 with Service Pack 1

Options:

Logon options: Automatic logon with current username and password

Help:

This policy setting allows you to manage settings for logon options.

If you enable this policy setting, you can choose from the following logon options.

Anonymous logon to disable HTTP authentication and use the guest account only for the Common Internet File System (CIFS) protocol.

Prompt for user name and password to query users for user IDs and passwords. After a user is queried, these values can be used silently for the remainder of the session.

Automatic logon only in Intranet zone to query users for user IDs and passwords in other zones. After a user is queried, these values can be used silently for the remainder of the session.

Automatic logon with current user name and password to attempt logon using Windows NT Challenge Response (also known as NTLM authentication). If Windows NT Challenge Response is supported by the server, the logon uses the user's network user name and password for logon. If Windows NT Challenge Response is not supported by the server, the user is queried to

Mozilla Firefox

Configuring Mozilla Firefox for Kerberos authentication

You can configure Microsoft Firefox browsers for Kerberos authentication.

Steps

1. Start Firefox.

2. Open a new tab and enter `about:config` in the address bar.

3. Double-click the `network.negotiate-auth.trusted-uris` preference name to modify its value to include the base URL of your PingFederate environment. For example, `www.example.com`.

4. Click **OK** and close the `about:config` tab.

Google Chrome

Configuring Google Chrome for Kerberos authentication

Google Chrome browsers support Kerberos authentication.

About this task

If you configure Microsoft Edge for Kerberos authentication, then you don't need to configure Google Chrome because Chrome uses the settings in Edge. For more information, see the Microsoft Edge tab on this topic.

OpenToken Adapter

To transfer identity and other user information between the PingFederate server and an end application, the PingFederate architecture allows for custom adapters to be deployed with the server.

PingFederate ships with a deployed OpenToken Adapter, which uses a secure token format `OpenToken` to transfer user attributes between an application and the PingFederate server.

On the identity provider (IdP) side, the OpenToken Adapter allows the PingFederate server to receive a user's identity from the IdP application.

For SAML connections, the IdP application can provide an authentication context to the service provider (SP) by including the `authnContext` attribute with the desired value in the secure token. Standard URIs are defined in the SAML specifications. For more information on assertions and protocol for SAML, see oasis-sstc-saml-core-1.1.pdf and saml-authn-context-2.0-os.pdf in the OASIS documentation.

If the secure token does not contain the `authnContext` attribute, PingFederate sets the authentication context as follows:

- `urn:oasis:names:tc:SAML:1.0:am:unspecified` for SAML 1.x
- `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` for SAML 2.0

As needed, the authentication context can be overridden by either an instance of the Requested AuthN Context Authentication Selector or the `SAML_AUTH_CTX` attribute in the SAML attribute contract. The latter takes precedence.

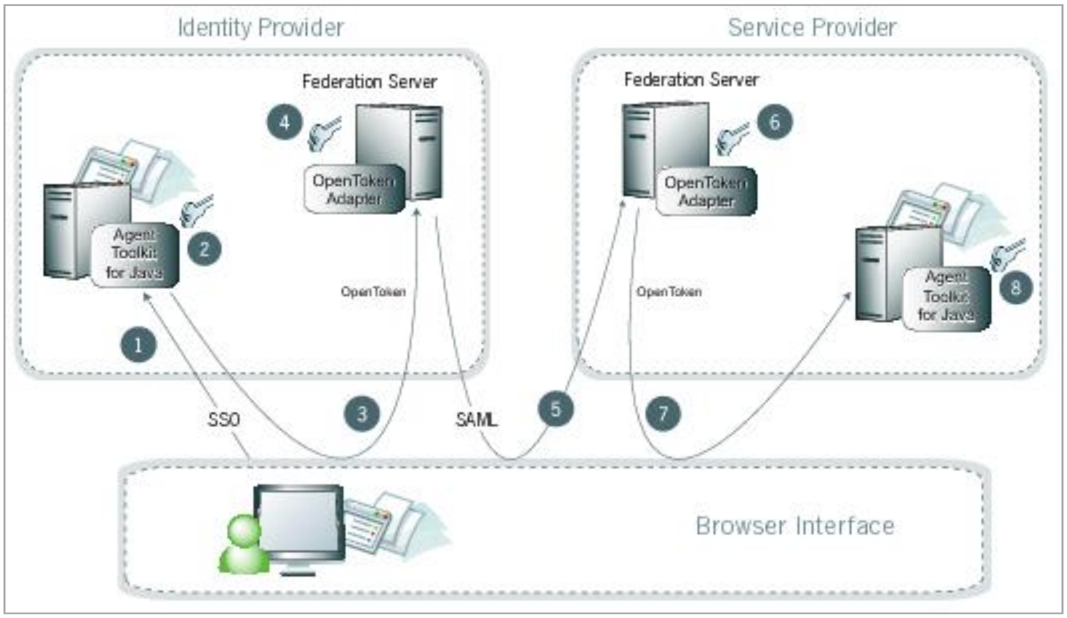
On the SP side, the OpenToken Adapter can be used to transfer user-identity information to the target SP application.

Specialized application integration kits are available from the Ping Identity Downloads website. Many kits leverage the OpenToken Adapter to integrate applications with the PingFederate server. The agent portions of the integration kits reside with the application and use the OpenToken to communicate with the OpenToken Adapter.

Note

To integrate applications for use with the OpenToken Adapter, download an integration kit for PingFederate from the Ping Identity Downloads website and follow instructions for installing and using Agent Toolkits in the accompanying documentation. Follow the configuration instructions in Configuring an OpenToken IdP Adapter instance to setup and to use with your applications.

The following figure shows a basic IdP-initiated single sign-on (SSO) scenario using PingFederate with the Java Integration Kit on both sides of an identity federation.



IdP-Initiated SSO: POST/POST

Processing steps

1. A user initiates an SSO transaction.
2. The IdP application inserts attributes into the Agent Toolkit for Java, which encrypts the data internally and generates an `OpenToken`.
3. A request containing the `OpenToken` is redirected to the PingFederate IdP server.
4. The server invokes the OpenToken IdP Adapter, which retrieves the `OpenToken`, decrypts, parses, and passes it to the PingFederate IdP server. The PingFederate IdP server then generates a SAML assertion.
5. The SAML assertion is sent to the SP site.
6. The PingFederate SP server parses the SAML assertion and passes the user attributes to the OpenToken SP Adapter. The Adapter encrypts the data internally and generates an `OpenToken`.
7. A request containing the `OpenToken` is redirected to the SP application.
8. The Agent Toolkit for Java decrypts and parses the `OpenToken` and makes the attributes available to the SP Application.

Configuring an OpenToken IdP Adapter instance

Configure an instance of the OpenToken IdP Adapter in PingFederate.

About this task

Configure an OpenToken Identity Provider (IdP) Adapter instance using the administrative console to enable a secure authentication plugin for your custom application.

Steps

1. Go to **Authentication > Integration > IdP Adapters**.

2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.

3. On the **Type** tab, configure the basics of this adapter instance:

    1. Enter the **Instance Name** and **Instance ID**.

    2. In the **Type** list, select the adapter type.

    3. (Optional) In the **Parent Instance** list, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

4. On the **IdP Adapter** tab, configure your OpenToken IdP Adapter instance.

    > **Note**
    >
    > These values depend on your developer's implementation.

    Learn more in the **Description** field provided on-window and in the following table.

    PingFederate's field names and descriptions for creating an OpenToken IdP Adapter instance

Field	Description
Password (Required)	The password to use for generating the encryption key. It is also known as the shared secret.
Confirm Password	Re-enter the password to use for generating the encryption key. It is also known as the shared secret.
Authentication Service (Required)	The URL to which the user is redirected for a single sign-on (SSO) event. This URL is part of an external application, which performs user authentication.
Click **Show Advanced Fields** in the **Instance Configuration** tab to review the following settings. Modify as needed.	
Transport Mode	How the token is transported to and from the application, either through a query parameter, a cookie (default), or as a form POST.
Token Name (Required)	The name of the cookie or query parameter that contains the token. This name must be unique for each adapter instance. Override the default value `opentoken` as needed.
Cipher Suite	The algorithm, cipher mode, and key size that should be used for encrypting the token. The default selected value is **AES-128/CBC**.
Logout Service	The URL to which the user is redirected for a single-logout event. This URL is part of an external application, which terminates the user session.
Cookie Demain	The server domain; for example, `example.com`. If no domain is specified, the value is obtained from the request.

Field	Description
Cookie Path	The path for the cookie that contains the token.
Token Lifetime (Required)	The duration in seconds for which the token is valid. Valid range is 1 to 28800. The default value is `300` (5 minutes).
Session Lifetime (Required)	The duration in seconds for which the token may be re-issued without authentication. Valid range is 1 to 259200. The default value is `43200` (12 hours).
Note Before Tolerance (Required)	The amount of time in seconds to allow for clock skew between servers. Valid range is 0 to 3600. The default value is `0`.
Force SunJCE Provider	If selected, the SunJCE provider is forced for encryption and decryption.
Use Verbose Error Messages	If selected, use verbose TokenException messages.
Obfuscate Password	If selected, the default, the password is obfuscated and password-strength validation is applied. Clearing the check box allows backward compatibility with previous OpenToken agents.
Session Cookie	If selected, OpenToken is set as a session cookie, rather than a persistent cookie. Applies only if the **Transport Mode** field is set as **Cookie**. The check box is not selected by default.
Secure Cookie	If selected, the OpenToken cookie is set only if the request is on a secure channel (https). Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Delete Cookie	If selected, the token cookie is deleted immediately after consumption. Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Replay Prevention	Selecting this option is recommended only if **Query Parameter** is the chosen token transport mode and form POST is used by an associated connection to send the SAML assertion. If selected, PingFederate ensures that the token can be used only once. By default, the check box is not selected. ⓘ **Note** Selecting this option might affect resource utilization and performance.
Skip Malformed Attribute Detection	If not selected, the default, it prevents insecure content from affecting the security of your application and the agent. Update your applications with the latest version of the agent. We recommend not to change the value of this flag.

5. On the **Actions** tab, click **Download** under **Action Invocation Link**, and then click **Export** to save the properties file.

   The values in the resulting file, `agent-config.txt`, represent the console configuration and are used by the Identity Provider (IdP) application. You can find more information in the documentation of your respective integration kit.

6. On the **Extended Contract** tab, configure additional attributes for this adapter instance as needed.

   The OpenToken IdP Adapter contract includes one core attribute: `subject`.

   The OpenToken IdP Adapter always extends the core contract with an attribute `userId` as well and fulfills it with the value of `subject` for backward compatibility reason.

7. On the **Adapter Attributes** tab, do the following:

   1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

      The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

      > **Note**
      >
      > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

      > **Important**
      >
      > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

   2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

      This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

      > **Note**
      >
      > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

   3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

      > **Note**
      >
      > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

   4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

8. (Optional) On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.

    - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
    - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

9. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

10. When finished in the **IdP Adapters** window, click **Save** to confirm the adapter instance configuration.

    If you want to exit without saving the configuration, click **Cancel**.

Configuring an OpenToken SP Adapter instance

Configure an instance of the deployed OpenToken Adapter, which uses a secure token format to transfer user attributes between an application and the PingFederate server.

About this task

Configure an OpenToken Service Provider (SP) Adapter instance to enable a secure transfer of the user-identity information to the target SP application.

Steps

1. Go to **Applications** > **Integration** > **SP Adapters** to access the **Manage SP Adapters Instances** window.

1. Click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.

2. On the **Type** tab, configure the basics of this adapter instance.

    1. Enter the **Instance Name**, **Instance ID**, and **Parent Instance** information and select the adapter type from the **Type** list.

       Select a **Parent Instance** from the list.This is useful when you are creating an instance that is similar to an existing instance. The child instance inherits the configuration of its parent. In addition, you have the option to override one or more settings during the rest of the setup. Select the **Override ...** check box and make the adjustments as needed in one or more subsequent windows.

3. On the **Instance Configuration** tab, configure your OpenToken SP Adapter instance security context.

    > **Note**
    >
    > These values are dependent on your developer’s implementation.

    For more information, see the **Description** field provided in-window and in the following table.

PingFederate's field names and descriptions for creating an adapter instance

Field	Description
Password Confirm Password (Required)	The password to use for generating the encryption key. It is also known as the shared secret.
Click **Show Advanced Fields** in the **Instance Configuration**tab to review the following settings. Modify as needed.	
Transport Mode	How the token is transported to and from the application, either through a query parameter, a cookie, or as a form POST (default).
Token Name	The name of the cookie or query parameter that contains the token. This name must be unique for each adapter instance. Override the default value `opentoken` as needed.
Cipher Suite	The algorithm, cipher mode, and key size that should be used for encrypting the token. The default selected value is **AES-128/CBC**.
Authentication Service	The URL to which the user is redirected for a single sign-on (SSO) event. This URL overrides the Target Resource, which is sent as a parameter to the Authentication Service.
Account Link Service	The URL to which the user is redirected for account linking. This URL is part of an external SP application. This external application performs user authentication and returns the local user ID inside the token.
Logout Service	The URL to which the user is redirected for a single-logout event. This URL is part of an external application, which terminates the user session.
Cookie Domain	The server domain; for example, `example.com`. If no domain is specified, the value is obtained from the request.
Cookie Path	The path for the cookie that contains the token.
Token Lifetime (Required)	The duration in seconds for which the token is valid. Valid range is 1 to 28800. The default value is `300` (5 minutes).
Session Lifetime (Required)	The duration in seconds for which the token may be re-issued without authentication. Valid range is 1 to 259200. The default value is `43200`, 12 hours.
Not Before Tolerance (Required)	The amount of time in seconds to allow for clock skew between servers. Valid range is 0 to 3600. The default value is `0`.
Force SunJCE Provider	If selected, the SunJCE provider is forced for encryption/decryption.
Use Verbose Error Messages	If selected, use verbose TokenException messages.

Field	Description
Obfuscate Password	If selected, the default, the password is obfuscated and password-strength validation is applied. Clearing the check box allows backward compatibility with previous OpenToken agents.
Session Cookie	If selected, OpenToken is set as a session cookie rather than a persistent cookie. Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Secure Cookie	If selected, the OpenToken cookie is set only if the request is on a secure channel (https). Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Send Subject as Query Parameter	Selecting this check box sends the user identifier `subject` as a clear-text query parameter, if the **Transport Mode** field is set to **Query Parameter**. If **Form POST** is the chosen token transport mode, the user identifier is sent as POST data.
Subject Query Parameter	The parameter name used for the user identifier when the **Send Subject ID as Query Parameter** check box is selected.
Send Extended Attributes	Extended Attributes are typically sent only within the token, but this option overrides the normal behavior and allows the attributes to be included in browser cookies or query parameters.
Skip Trimming of Trailing Backslashes	If not selected, the default, it prevents insecure content from affecting the security of your application and the agent. Update your applications with the latest version of the agent. We recommend not to change the value of this flag to avoid a negative security impact, such as someone maliciously adding slashes to exploit the system.
URL Encode Cookie Values	If checked, the extended attribute cookie value will be URL encoded.

4. In the **Actions** tab, click **Download** under **Action** section. Click **Export** to save the properties file.

   The values in the resulting file, `agent-config.txt`, represent the console configuration and are used by the SP application. See the documentation of your respective integration kit for more information.

5. **Optional:** In the **Extended Contract** tab, configure additional attributes for this adapter instance.

6. In the **Summary** tab, review your configuration, modify as needed. Click **Done**.

7. On the **SP Adapters** window, click **Save** to confirm the adapter instance configuration.

   If you want to exit without saving the configuration, click **Cancel**.

Configuring a Passthrough IdP Adapter

The Passthrough IdP Adapter allows a user key to be associated with a user's authentication sessions.

About this task

By placing the Passthrough IdP Adapter downstream from an IdP connection in a policy tree, you can take advantage of additional capabilities associated with defining a user key. For example, you can use the user key to query or revoke a user's authentication sessions. The adapter automatically sets the `username` attribute in its core contract to match the configured incoming user ID. With an upstream IdP connection, the incoming user ID can be mapped to the connection's `SAML_SUBJECT` attribute.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select **Passthrough IdP Adapter**.
    3. **Optional:** In the **Parent Instance** list, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.
4. On the **IdP Adapter** tab, there are no configurable fields, so skip to the **Extended Contract** tab.
5. On the **Extended Contract** tab, you can extend the contract by entering the name of the desired attribute and clicking **Add**. You can add multiple attributes.
6. On the **Adapter Attributes** tab, do the following:
    1. **Optional:** From the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

       The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

       > **Note**
       >
       > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validators to trim spaces.
    2. Select the check box under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

       This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

       > **Note**
       >
       > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

3. Select the check box under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

> **Note**
>
> Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

4. Select the **Mask all OGNL-expression generated log values** check box if OGNL expressions might be used to map derived values into outgoing assertions and you want those values masked.

7. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.
   - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
   - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

8. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

Configuring a Reference ID Adapter

The Reference ID Adapter allows user attributes to be passed in and out of the PingFederate server through direct HTTP(S) calls. Attributes are retrieved using a Reference ID.

Steps

1. Go to **Authentication > Integration > IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
   1. Enter the **Instance Name** and **Instance ID**.
   2. In the **Type** list, select the adapter type.
   3. Optional: In the **Parent Instance** list, select an existing type.

      If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. On the **IdP Adapter** tab, configure the Reference ID Adapter instance as follows:
   1. Enter values for the adapter configuration, as described in the following table.

Field	Description
Authentication Endpoint	The application endpoint URL where the end user is redirected for authentication.
User Name	The ID that the application uses to authenticate to the PingFederate server.
Pass Phrase	The pass phrase that the application uses to authenticate to the PingFederate server.
Allowed Subject DN	The subject DN from the client certificate. If entered, PingFederate restricts client-certificate authentication (when enabled) by matching against this DN. **Note** This field supports the asterisk (`\*`) wildcard character and multiple DNs, separated by the pipe '|'.
Allowed Issuer DN	The issuer DN from the client certificate. If entered, PingFederate restricts client-certificate authentication (when enabled) by matching against this DN. **Note** Supports the asterisk (`\*`) wildcard character and multiple DNs, separated by the pipe '|'.
Logout Service Endpoint	The application endpoint URL used for single logout. The **Logout Service Endpoint** works in conjunction with **Logout Mode**.
Logout Mode	Select the option that determines how the application logout is handled. **Front Channel** - Redirects the user to the application endpoint and expects the application to redirect back to the provided PingFederate resume path. **Back Channel** - Sends a direct HTTP request from the server to the application. The default setting is **None**.

    2. **Optional:** Click **Show Advanced Fields** to review or modify default values.

2. On the **Actions** tab, you can optionally click **Show Pass Phrase** to display the pass phrase for the adapter.

3. On the **Extended Contract** tab, extend the contract as needed by entering the name of the desired attribute and clicking **Add**. You can add as many attributes as needed.

4. On the **Adapter Attributes** tab, do the following:

    1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

       The attribute’s value is used to identify user sessions across all adapters. **None** is selected by default.

> **Note**
>
> If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

> **Important**
>
> For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

   This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

   > **Note**
   >
   > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

   > **Note**
   >
   > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

5. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.
   - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
   - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

6. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

Configuring an X.509 Certificate IdP Adapter

The X.509 Certificate IdP Adapter allows a PingFederate identity provider (IdP) server to perform client X.509 certificate authentication for single sign-on (SSO) to service provider (SP) applications.

About this task

When PingFederate is acting as an IdP, this adapter validates X.509 certificates against the certificate authority (CA) in the PingFederate certificate store. Use the **IdP Adapter** window to configure the X.509 Certificate IdP Adapter.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. On the **IdP Adapters** page, click **Create New Instance** to start the **Create Adapter Instance** configuration.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.
    3. Optional: In the **Parent Instance** list, select an existing type.

       If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. **Optional:** On the **IdP Adapter** tab, in the **Constrain Acceptable Root Issuers** section, specify the CAs that you want to use to validate end-user X.509 certificates.

   > **Note**
   >
   > Client certificates are always validated against all trusted CAs in PingFederate and the Java Virtual Machine. This section only restricts which issuers are used to validate end-user certificates.

    1. Click **Add a new row to 'Constrain Acceptable Root Issuers'**.
    2. In the **Issuer DN** field, enter the subject distinguished name (DN) of an issuer listed on the **Trusted CAs** window. For more information, see Manage trusted certificate authorities.
    3. In the **Action** column, click **Update**.
    4. To add more acceptable issuers, repeat steps a-c.

2. Enter values for the adapter configuration, as described in the following table.

Field	Description
Client Auth Port	The port that PingFederate uses to validate client certificates.
Client Auth Hostname	The PingFederate hostname that is configured to use client-certificate authentication.
Parse Client Cert Subject and Issuer DNs	When enabled, the subject and issuer distinguished name (DN) in the client certificate are treated as separate attributes. This allows you to do the following: ◦ Add subject or issuer DN attributes, such as CN or UID, to the adapter's extended contract. ◦ Use the subject DN `email` attribute in the adapter's core contract. ◦ Use Object-Graph Navigation Language (OGNL) expressions to extract other information from the X.509 certificate. This option is enabled by default.
Match Issuer DN in Client X.509 Certificate	Determines how PingFederate validates the issuer distinguished name (DN) for the client certificate. When selected, the issuer DN is matched against the entries that are defined in the **Constrain Acceptable Root Issuers** section. When cleared, the issuer DN is matched against the default top level certificate in the chain that is presented by the client.
Advanced Fields	
Return Success on SLO	When enabled, a "success" message is sent in response when the adapter receives a single logout (SLO) request. SLO is not supported by this adapter and the user session is not terminated. This feature only prevents other sites from experiencing an SLO failure. This option is enabled by default.
Authentication Context	The value used to populate the "Authentication Context" field in the Security Assertion Markup Language (SAML) token that PingFederate sends after validating the X.509 certificate. **Default** - Sets the value to "TLSClient". **Policy OID** - Sets the value to the identifier for the policy. **Custom** - Sets based on the value you enter in the **Custom Authentication Context** field.
Custom Authentication Context	The value used to populate the "Authentication Context" field in the SAML token. Applies when **Authentication Context** is set to **Custom**.

Field	Description
Include Subject Alternative Name (SAN)	When enabled, the adapter includes the following decoded SAN attributes from the X.509 certificate and makes them available in the attribute contract: ◦ userPrincipalName ◦ RFC822Name ◦ fascn_sen ◦ fascn_wo_sen ◦ fascn_hex ◦ deviceId

3. On the **Extended Contract** tab, add any attributes, that you want to include in the extended contract. Enter attributes in uppercase. Only attributes specified in RFC 2253 are allowed: `CN`, `L`, `ST`, `O`, `OU`, `C`, `STREET`, `DC`, and `UID`.

   > **Note**
   >
   > You can include subject DN components in this list.
   > If you selected **Parse Client Cert Subject and Issuer DNs** on the **IdP Adapter** tab, you can also include the subject DN `email` component, as well as issuer DN components.
   > For issuer DN components, prefix the attribute with `issuer_`, such as `issuer_CN`.

4. On the **Adapter Attributes** tab, do the following:

   1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

      The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

      > **Note**
      >
      > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

      > **Important**
      >
      > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

   2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

      This selection is used if any of your SP partners use pseudonyms for account linking.

> **Note**
>
> A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

> **Note**
>
> Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

5. **Optional**: On the **Adapter Contract Mapping** tab, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for datastore queries.
   - Fulfill adapter contract with values from the adapter, the default, datastore queries, if configured, context of the request, text, or expressions, if enabled.
   - Set up the Token Authorization framework to validate one or more criteria prior to the issuance of the adapter contract.

6. On the **Summary** tab, review your configuration and modify as needed. Click **Save**.

Customer IAM configuration

PingFederate empowers administrators to deliver a secure and easy-to-use customer authentication, registration, and profile management solution. This solution leverages the HTML Form Adapter to offer users the options to authenticate through third-party identity providers (IdPs), self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page.

Like other user-facing windows, administrators can customize and localize both the registration and profile management pages to present a consistent branding experience based on the needs of the users and the organizations.

Furthermore, administrators can allow users to leverage their existing identities from third-party identity providers. Any IdP connection or IdP adapter, such as the LinkedIn Cloud Identity Connector, can be used as an authentication source to a third-party identity provider. This optional capability enables a mapping configuration between the attributes returned by the identity provider and the fields within the registration page, streamlining the registration process.

Depending on the requirements and configuration of existing components, the configuration process might involve changes to these configuration components: authentication policy contracts, local identity profiles, HTML Form Adapter instances, and IdP authentication policies.

> **Note**
>
> The HTML Form Adapter is authentication API-capable. The PingFederate authentication API is a JSON-based API that enables end-user interactions, such as credential prompts, to be handled by an external web application. This API does so by providing access to the current state of the flow as an end user steps through a PingFederate authentication policy. For more information, see Authentication applications and the authentication API.

Setting up PingDirectory for customer identities

PingFederate can optionally store customer identities in PingDirectory. After you have installed PingDirectory, update the LDAP schema with a new object class and a couple attributes to store customer identities and their connections.

About this task

Update the LDAP schema with a new object class and a couple attributes using an LDIF file provided. To optimize performance, apply updates in indexes to the directory as well. In addition, you must configure in PingFederate an LDAP datastore connection to your PingDirectory and an LDAP Username password credential validator (PCV) instance for the HTML Form Adapter to validate user credentials. If you have previously created these components, you can reuse them.

> **Note**
>
> Skip this configuration if your use case does not involve registration or profile management. For more information, see Enabling third-party identity providers without registration.

Steps

1. Update the LDAP schema.

    1. Sign on to the PingDirectory administrative console.

    2. Go to the **LDAP Schema > Schema Utilities** screen.

    3. Click **Import Schema Element**.

    4. Copy the schema changes from the `<pf_install>/pingfederate/server/default/conf/local-identity/ldif-scripts/local-identity-pingdirectory.ldif` file and paste them into the text area.

        If you are creating a new organizational unit as part of the LDIF import, edit the DN information.

    5. Click **Import**.

2. Create an equality index for the `pf-connected-identity` attribute.

    Use PingDirectory's **`dsconfig`** utility to create this index. The **`dsconfig`** utility is interactive. You can also provide inputs as command arguments. For example, the following samples create the `pf-connected-identity` index.

```
$ bin/dsconfig create-local-db-index \
                 --backend-name userRoot \
                 --index-name pf-connected-identity \
                 --set index-type:equality
```

After adding the index, use the `rebuild-index` utility to build the indexes. For instance, the following sample builds the required index.

```
$ bin/rebuild-index \
                          --baseDN "dc=example,dc=com" \
                          --index pf-connected-identity
```

3. Create an LDAP datastore connection to your PingDirectory on **System > Data Stores**.

   If you have already created an LDAP datastore connection to your PingDirectory, you can reuse it.

   > **Note**
   >
   > When configuring the LDAP datastore connection to PingDirectory, you specify your service account with a Bind distinguished name (DN) and password. This service account requires specific access control instruction (ACI)s in PingDirectory to enable PingFederate's Local Identity Profile (LIP) functionalities.
   > The specific ACIs depend on which LIP features you enable, like user creation, authentication, password reset, profile updates, or account unlock.
   > You can find detailed ACI examples and best practices in the following Ping Identity Support article: Configuring PingDirectory and PingFederate for Self-service Password Reset.

4. Create an instance of the LDAP Username PCV on **System > Password Credential Validators** to validate user credentials stored in PingDirectory.

   If you have already created an LDAP Username PCV instance, you can reuse it.

   > **Note**
   >
   > Later you will create a local identity profile as part of the customer IAM configuration. The **Search Base** value here should match the **Base DN** value defined in the local identity profile. For more information, see Configuring LDAP base DN and attributes.

5. For the local identity profile management page to work correctly, enable PingDirectory's `ds-pwp-state-json` virtual attribute.

   Use PingDirectory's `dsconfig` utility and a command like the following to enable the `ds-pwp-state-json` virtual attribute for users with the object class `person`:

```
dsconfig set-virtual-attribute-prop \
   --name "Password Policy State JSON" \
   --set enabled:true \
   --set require-explicit-request-by-name:true \
   --set "filter:(objectClass=person)"
```

Related links

- Installing the Server
- Using the Schema Editor Utilities
- Working with Indexes
- Configuring an LDAP connection
- Configuring the LDAP Username Password Credential Validator

Managing local identity profiles

A local identity profile (LIP) is a stored user identity (PingDirectory) created and maintained by PingFederate. It provides the capability for user creation and administration, and centralizes those policies with the authentication and authorization policies already within PingFederate.

About this task

Users can enter their information during registration using a link on the HTML Login page or after successful authentication from a third-party IdP. You can configure LIPs for user registration, third-party federation (provisioned through SAML), and user profile management.

A typical customer identity and access management (CIAM) use case only requires one LIP. As needed, you can create multiple profiles to suit the needs of your organization. Using the administrative console, LIPs are defined in the **Identity Policies** section.

> **Note**
>
> As of PingFederate 10.1, an authentication session is automatically created for a user after registration, preventing the user from having to log in again during the next single sign-on (SSO) transaction. This feature is enabled by default for all new and existing local identity profiles. However, if needed, you can disable it through the `/localIdentity/identityProfiles` administrative API endpoint by setting the `createAuthnSessionAfterRegistration` attribute to `false`.
> When associated with an HTML Form Adapter instance, a local identity profile provides users the option to authenticate through third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page.

Steps

- To configure a new profile, go to **Authentication** > **Policies** > **Local Identity Profiles**. Click **Create New Profile**.
- To modify an existing profile, select it by its name under **Local Identity Profile Name**.
- To review the usage of an existing profile, click **Check Usage** under **Action**.
- To remove an existing instance or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

Next steps

See the following topics for detailed instructions that enable you to complete these configurations:

- Setting up self-service registration
- Enabling third-party identity providers

- Enabling profile management
- Creating advanced registration mapping
- Enabling third-party identity providers without registration

Configuring local identity profiles

In the **Local Identity Profiles** window, configure a new local identity profile and provide profile information on the **Profile Info** tab.

About this task

This section only provides instructions for configuring the fields in the Local Identity Profile (LIP) configuration tabs. For instructions on using the LIP configuration as part a greater customer identity access management solution, see the following topics:

- Setting up self-service registration
- Enabling third-party identity providers
- Enabling profile management
- Creating advanced registration mapping
- Enabling third-party identity providers without registration

Steps

1. To configure a new profile, go to **Authentication** > **Policies** > **Local Identity Profiles**. Click **Create New Profile**.
2. On the **Profile Info** tab, in the **Local Identity Profile Name** field, enter a name.
3. From the **Authentication Policy Contract** list, select a contract.

   If you have not yet defined the desired contract, click **Manage Policy Contracts**.
4. Select the **Enable Registration** check box if you want to enable users to complete a self-service registration as part of the sign-on experience through an instance of the HTML Form Adapter.
5. Select the **Enable Profile Management** check box if you want to allow users to manage their accounts.
6. When finished, click **Next**.

Defining authentication sources

Authentication sources are identifiers for third-party identity providers, such as social providers used to display these providers on the HTML form adapter user interface as alternate authentication and registration options. They are also used in authentication policies to configure branches to identity provider (IdP) adapters and connections.

About this task

Authentication sources are optional. They are the identifiers for third-party identity providers, such as social network providers. When defined, the associated HTML Form Adapter instance displays them on the sign-on page as alternative options for authentication and registration, if enabled. If profile management is enabled, users can connect or disconnect third-party identity providers to and from their accounts.

You can store attributes received from third-party identity providers as part of the user records. If required, attributes can be updated as users authenticate. By default, attributes are removed from user records as users disconnect third-party identity providers from their accounts. It is worth noting that storing attributes received from third-party identity providers is optional and configurable on a per-local identity profile basis. Additionally, this option is only applicable when a local identity profile is configured with registration, profile management, or both.

Steps

1. On the **Authentication Sources** tab, type a source in the **Authentication Source** field, and click **Add**.

   > **Tip**
   >
   > If you use the authentication source names Facebook, Google, LinkedIn, Twitter, FIDO, the HTML Form Adapter default templates render the associated icons on the registration and profile management pages.

   > **Note**
   >
   > As of PingFederate 10.2, you can use `Security Key` as an authentication source. The Security Key authentication source automatically adds a Security Key button to the HTML Form Adapter and Local Identity Profile management and registration pages. It allows users to authenticate with a hardware security key such as YubiKey. The button displays only if the user's device or browser supports the Web Authentication (WebAuthn) protocol.
   > You can also use the Security Key authentication source with the PingID adapter by configuring a policy tree with a rule that includes a `policy.action` attribute equal to a **Value** and **Result** of `Security Key`. Then select the PingID Adapter under **Security Key** in the policy and configure it as needed. For information about configuring authentication policies and rules, see Defining authentication policies and Configuring rules in authentication policies.

   Result:

   Make a note of the values defined here. In a later step, you will create a rule for each authentication source in an identity provider (IdP) authentication policy. Each rule forms a policy path that initiates the authentication process.

2. If needed, modify or remove existing authentication sources.

   > **Caution**
   >
   > When removing an authentication source, keep in mind that accounts that were created using the associated third-party identity provider will no longer be usable after the removal. To minimize the risk of accidental removals, the administrative console prompts to confirm each removal request.

3. Configure storage settings for attributes received from third-party IdP.

   > **Note**
   >
   > The attribute storage settings are inapplicable and not shown if neither **Enable Registration** nor **Enable Profile Management** check box not selected on the **Profile Info** tab.

Choose from:

- If storing attributes, select the **Store Attributes** check box.
- If you want attributes retained after users disconnect third-party IdP from their accounts, select the **Keep Attributes After Users Disconnect** check box.
- If you want attributes updated as users authenticate, select the **Update Attributes When Users Authenticate** check box and enter a value in the **Minimum Number of Days Between Updates** field.

4. When finished, click **Next**.

Related links

- Enabling third-party identity providers
- Enabling third-party identity providers without registration
- Troubleshooting registration and profile management issues

Configuring local identity fields

Configuring local identity fields determines which fields will be displayed as input controls on the user registration and profile management pages.

About this task

When registration is enabled for a local identity profile, select a local identity field to be the unique identifier for the purpose of identifying the users. To enable email ownership verification, add a field to store the email address and another field to store the verification status. While the former can be any field that uses the **Email** or **Text** input control, the latter must use the **Hidden** input control.

Steps

1. On the **Fields** tab, click **Create New Field**.
2. In the **ID** field, enter a unique identifier. The ID references the field throughout the user interface and is the field name in the HTML template.
3. In the **Label** field, type the field name that users see on the user registration and profile management pages.
4. From the **Type** list, select the type of input control for the field being configured.
5. Under **Applies To**, select one or both options to configure this field to appear on the user registration page, the profile management page, or both. Both options are selected by default.

   This step is applicable only if both **Enable Registration** and **Enable Profile Management** are enabled on the **Profile Info** tab.
6. **Optional:** Select the relevant parameters under **Parameters**.

   You can make a non-hidden field required or read-only. You can also configure PingFederate not to record values from this field in logs.
7. **Optional:** Enter a value under **Default Value**.

Specifying a default value can streamline the registration process. This is the default value of the field unless another value is specified in the authentication policy. For more information, see Configuring local identity mapping.

Default Value is not shown if you have chosen an input control of **Checkbox group**, **Email**, **Phone**, or **Hidden**, or the **Read-Only** parameter.

8. Add the applicable predefined values under **Options**.

   This step is applicable and required only if you have chosen **Checkbox Group** or **Dropdown** as the input control.

9. Click **Done**.

   Result:

   The administrative console returns to the **Fields** tab.

10. Select the **Unique ID** option for the applicable field. This is applicable and required only if registration is enabled on the **Profile Info** tab.

    > **Note**
    >
    > You cannot choose any field that uses the **Checkbox**, **Checkbox Group**, **Date**, or **Dropdown** input control as a unique identifier, because values from these fields will likely collide as the population of users grows.

11. Clear the check box beside **Strip Leading/Trailing Spaces From the Value of the Unique ID Field** only if you do not want to check for leading and trailing spaces in the unique ID field.

12. Select **Mask All OGNL-Expression Generated Log Values** if you want to mask local identity field values in logs when OGNL expressions might be used to map derived values into outbound single sign-on (SSO) tokens in authentication policies.

13. Click **Next**.

Configuring email ownership verification options

Based on your customer IAM use cases, you can optionally offer users the opportunity to confirm the ownership of the email address associated with their accounts. This configuration can be configured on a per-local identity profile basis.

About this task

Using the administrative console, configure the email ownership verifications settings for a local identity profile.

If you enable email ownership verification, when a user submits a registration request, PingFederate sends an email ownership verification message to the email address. The message is valid for a configurable period. If the user cannot find the message, they can request another one by accessing the email ownership verification endpoint. Moreover, if you enable profile management, the profile management page displays a reminder until the user verifies the email address. Like other local identity fields, the email verification status is stored in the directory and can be relayed to the applicable target applications through identity provider (IdP) authentication policies.

Steps

1. Go to **Authentication** > **Policies** > **Local Identity Profiles**.

2. On the **Email Verification** tab, select the **Enable Email Ownership Verification** check box to offer users the opportunity to verify the email address associated with their accounts.

   The **Email Verification** tab appears only when you select the **Enable Registration** check box or the **Enable Profile Management** check box on the **Profile Info** tab.

   The **Enable Email Ownership Verification** check box is not selected by default.

   > **Note**
   >
   > The rest of the steps apply only if you select to enable email ownership verification.

3. In the **Email Address Field** list, select a field.

   The field value represents the recipient of the verification message.

   Only fields that use the **Email** or **Text** input control are eligible and shown.

4. In the **Ownership Status Field** list, select a field.

   The field value represents the email ownership verification status. PingFederate sets the value to `false` in the directory when it receives a new or an updated email address from the user. After the user verifies the email ownership, PingFederate sets the value to `true`.

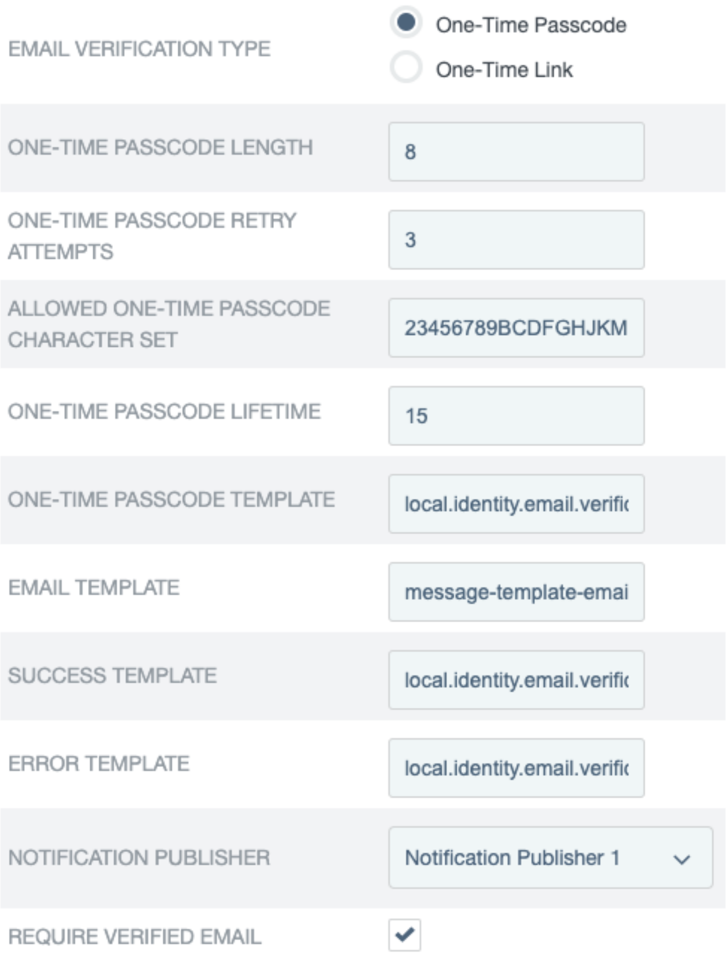
   Only fields that use the **Hidden** input control are eligible and shown.

5. To verify emails with one-time passcodes, which is the default option:

   1. Set the **Email Verification Type** to **One-Time Passcode**.

   2. (Optional) Change any of the values in the following fields:

      - **One-Time Passcode Length**
      - **One-Time Passcode Retry Attempts**
      - **Allowed One-Time Passcode Character Set**
      - **One-Time Passcode Lifetime**

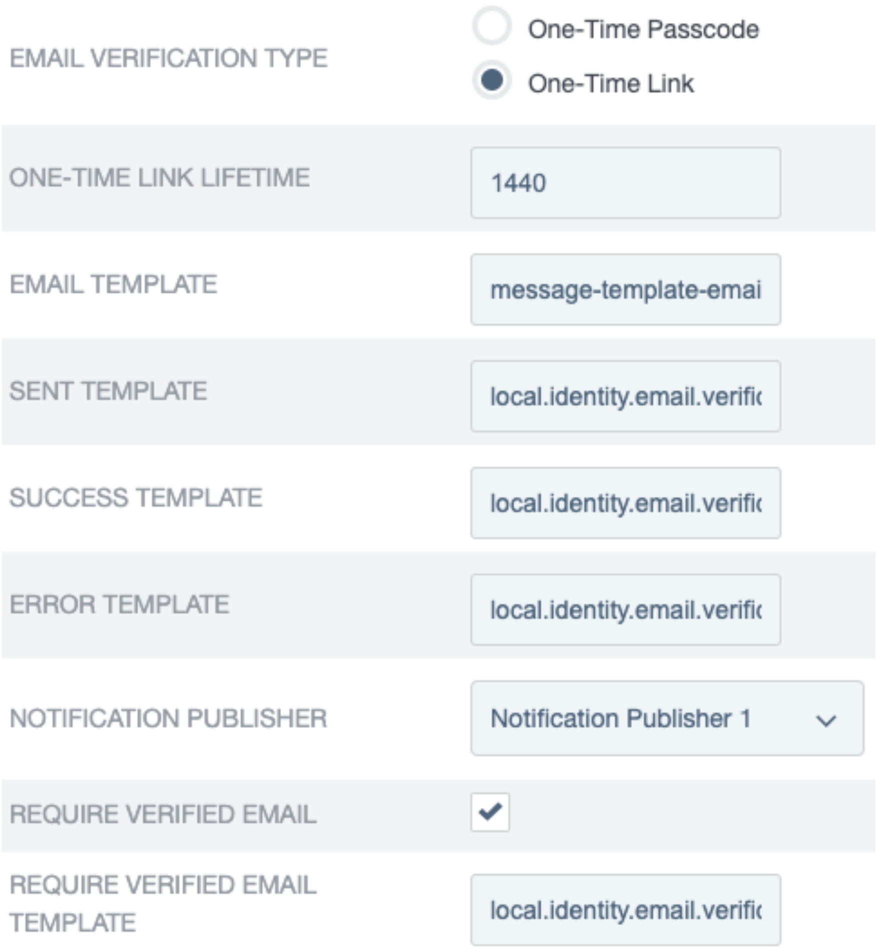


EMAIL VERIFICATION TYPE	● One-Time Passcode ○ One-Time Link
ONE-TIME PASSCODE LENGTH	8
ONE-TIME PASSCODE RETRY ATTEMPTS	3
ALLOWED ONE-TIME PASSCODE CHARACTER SET	23456789BCDFGHJKM
ONE-TIME PASSCODE LIFETIME	15
ONE-TIME PASSCODE TEMPLATE	local.identity.email.verifi
EMAIL TEMPLATE	message-template-emai
SUCCESS TEMPLATE	local.identity.email.verifi
ERROR TEMPLATE	local.identity.email.verifi
NOTIFICATION PUBLISHER	Notification Publisher 1
REQUIRE VERIFIED EMAIL	☑

6. To verify emails with one-time links:

   1. Set the **Email Verification Type** to **One-Time Link**.

   2. (Optional) Change the length of the **One-Time Link Lifetime**.



EMAIL VERIFICATION TYPE	○ One-Time Passcode ● One-Time Link
ONE-TIME LINK LIFETIME	1440
EMAIL TEMPLATE	message-template-emai
SENT TEMPLATE	local.identity.email.verific
SUCCESS TEMPLATE	local.identity.email.verific
ERROR TEMPLATE	local.identity.email.verific
NOTIFICATION PUBLISHER	Notification Publisher 1
REQUIRE VERIFIED EMAIL	☑
REQUIRE VERIFIED EMAIL TEMPLATE	local.identity.email.verific

7. (Optional) To use different template files, update the template fields.

   The template files are in the `<pf_install>/pingfederate/server/default/conf/template` directory and the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory.

   > **Note**
   >
   > The email templates are only applicable when using an SMTP notification publisher to deliver email-verification messages.

8. Select a **Notification Publisher** instance.

   If you haven't yet configured the desired notification publisher instance, click **Manage Notification Publishers**, configure the instance, and then select it.

9. (Optional) To require users to verify their email address ownership before they can access any connected applications:

   1. Select the **Require Verified Email** check box.
   2. (Optional) If the **Email Verification Type** is set to **One-Time Link**, you can specify a different template in the **Require Verified Email Template** field.

      The default template gives users options to **Resend** the verification email, **Continue**, or **Cancel**.

      The **Require Verified Email Template** field appears only after you select the **Require Verified Email** check box.

> **Note**
>
> If you select the **Require Verified Email** check box, users can sign on to their local identity profile and manage their account but PingFederate blocks them from accessing any connected applications until they have successfully verified their email address.
> To let users manage their accounts, select the **Enable Profile Management** check box on the **Profile Info** tab, as described in Configuring local identity profiles.

10. Click **Next**.

Configuring registration options

On the **Local Identity Profile** page's **Registration** tab, you can configure the user registration experience and specify the template file for the registration page.

Steps

1. Go to **Authentication** > **Policies** > **Local Identity Profiles** > **Registration**.
2. If you want to enable reCAPTCHA from Google to prevent automated registration attempts:
    1. Select the **Risk Enabled** checkbox.
    2. Select a **Risk Provider**. The default selection is **Default**, which is the service provider specified as the default on the **CAPTCHA and Risk Providers** page.

    > **Note**
    >
    > To add a risk provider instance, click **Manage CAPTCHA and Risk Providers** to open the **CAPTCHA and Risk Providers** page, then follow the steps in Managing CAPTCHA and risk providers.
3. To use a different template file, update the **Registration Template** field. The default value is `local.identity.registration.html`.
4. To allow users to indicate whether their device is shared or private, select the **Enable 'This is my device'** checkbox.
5. For PingFederate to create an authentication session after a local account is registered, leave the **Create Session After Registration** checkbox selected. It is selected by default.
6. To override the value in the **Unique ID** field as the username that is sent to adapters in the policy, select a username in the **Username Field** list.
7. If you have a policy fragment that needs to be executed as part of the workflow, select the fragment in the **Registration Workflow** list. Click **Add Policy Fragments**, if needed, to create one or more fragments.

    When you select a fragment, you can then choose whether PingFederate should execute the registration workflow before or after account creation. **After Account Creation** is selected by default.

> **Note**
>
> The registration fragment always executes after the user has entered their information in the registration form. There’s an implicit mapping between fields in the local identity profile (LIP), which may have been populated in the registration form, and attributes in the registration fragment input authentication policy contract (APC). Implicit mapping is a mapping that executes automatically if the name of a field in the LIP matches the name of an attribute in the APC.
> There’s also an implicit mapping between attributes in the registration fragment output APC and fields in the LIP. So, you can use the fragment to populate or overwrite the LIP fields. This works whether you configure the fragment to execute before or after account creation.

8. Click **Next**.

Related links

- Customizable user-facing pages
- Localizing messages for end users

Configuring profile management options

Configure the profile management experience and specify the template file for the profile management page.

About this task

Using the administrative console, configure the user registration in the local identity profiles section.

Steps

1. Go to **Authentication** > **Policies** > **Local Identity Profiles**. In the **Local Identity Profile** window, using the tabs, configure the user registration page settings.

   Choose from:

   - If you want to give users the option to delete their local accounts without administrator assistance, select the **Enable Profile Deletion** check box.

     This check box is not selected by default.

     If enabled, when users choose to delete their accounts, their user records are removed from your directory.

   - If you want to use a different template file, update the **Registration Template** field.

   The default value is `local.identity.profile.html`.

Related links

- Customizable user-facing pages
- Localizing messages for end users

Managing datastore configuration

PingFederate requires datastore configuration because it stores customer identities in PingDirectory.

Configure the datastore where local identities are stored.

- To begin, click **Configure Data Store**.

Selecting a datastore for customer identities

PingFederate stores customer identities in PingDirectory, improving the flexibility, scale, and security of your user data. Follow these steps to connect PingFederate to a directory to store identity and profile data and expose the data to all applications and channels through LDAPv3.

About this task

Using the **Data Store Configuration** tab, connect your LDAP datastore to PingFederate.

Steps

1. Go to **Authentication** > **Policies** > **Local Identity Profiles**.
2. On the **Data Store Configuration** tab, click **Configure Data Store**.

   Result:

   This will open a **Data Store** window.
3. On the **Data Store Type** tab, enter a name in the **Name** field, and from the **Type** list, select **Directory (LDAP)**.
4. Click **Next.**

   If you have not yet created an LDAP datastore to connect PingFederate to your PingDirectory or if you want to review your LDAP datastore settings, click **Manage Data Store**.

Related links

- Setting up PingDirectory for customer identities

Configuring LDAP base DN and attributes

Configure the datastore to search for a user's authentication starting with the base distinguished name (DN) and attributes within the LDAP directory.

About this task

On the **LDAP Configuration** tab, specify the branch of your directory hierarchy where you want PingFederate to store customer identities. Then, select the object class and the attributes to be associated with local identity fields.

> **Note**
>
> Later you will associate the local identity profile with an HTML Form Adapter instance and apply the profile in an identity provider (IdP) authentication policy as part of the customer IAM configuration. If your use case requires registration or profile management, the policy engine must look up the users as they access the registration page or the profile management page. The scope of this search begins at the base DN defined here.
> For this reason, the base DN here should match the value of the **Search Base** field defined in the LDAP Username Password Credential Validator instance used by the associated HTML Form Adapter instance.

For more information about each field, refer to the following table.

Field	Description
Base DN	The base distinguished name of the tree structure where PingFederate stores customer identities.
Root Object Class	The object class containing the desired attributes.
Attributes	A list of attributes based on the selected **Root Object Class** value.

Steps

1. Go to **Authentication** > **Policies** > **Local Identity Profiles**.
2. On the **Data Store Configuration** tab, click **Configure Data Store**.

   Result:

   This will open a **Data Store** window.
3. On the **LDAP Configuration** tab, enter the applicable fields.
4. In the **User DN** field, specify a base DN.

   > **Tip**
   >
   > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.
5. **Optional:** Click **View Local Identity Fields** to determine which attributes from the directory server should be added to the local identity profile.
6. From the **LDAP Configuration** tab, click **Advanced**.

   Result:

   This will open the **LDAP Binary Attributes** tab.
7. On the **LDAP Binary Attributes** tab, add attributes.
   1. Enter a name in the **Binary Attribute Name**.
   2. Click **Add**.

3. Select a root object class, select an applicable attribute, and then click **Add Attribute**.

   Repeat this step to add more attributes as needed.

8. Click **Done**. Click **Save**.

Related links

- Setting up PingDirectory for customer identities

Configuring LDAP relative DN and object class

When a user submits a registration request, PingFederate formulates the distinguished name (DN) of the user by prefixing the relative distinguished name (RDN) to the base DN defined in the LDAP configuration and then asks PingDirectory to create a new account based on the selected object class.

Steps

1. **Optional:** Click **View List of Available LDAP Attributes** to determine which LDAP attributes can be used to construct the RDN pattern.
2. In the **Relative DN Pattern** field, enter a valid pattern.

   The pattern is as follows.

   `[.parmname]` attribute1 `=value1…,` `[.parmname]` attributeN `=valueN]`

   If you want to use the *${entryUUID}* variable to guarantee the uniqueness of the relative DNs for all users, you must use it with the `entryUUID` LDAP attribute, such as in the following example.

   `entryUUID=${entryUUID}`
3. From the **Object Class** list, select the primary objectClass value used when creating a new local identity profile; for example, inetOrgPerson.
4. **Optional:** From the **Auxiliary Object Class Name** list, select an objectClass that contains additional required attributes that are not available in the primary objectClass, and click **Add**. You can add multiple object classes as needed.

   For example, if you require the placeOfBirth attribute for a user's profile, you can add the naturalPerson root object class to the **Auxiliary Object Class Name** list. Doing this requires that you have done the following:

   1. Added placeOfBirth as a field on the **Fields** tab when configuring the local identity profile.
   2. Added the naturalPerson root object class and placeOfBirth attribute on the **LDAP Configuration** tab when configuring the datastore.
5. Click **Next**.

Related links

- https://docs.pingidentity.com/bundle/pingdirectory-81/page/xpb1564011435707.html*PingDirectory Administration Guide*]

Defining datastore mapping configuration

Configure the mapping between the local identity profile fields and the datastore attributes.

About this task

Steps

1. In the **Data Store Configuration**, on the **Data Store Mapping** tab, select an LDAP attribute under **Data Store Attribute** for each local identity field.

Reviewing datastore configuration

On the **Summary** tab, you can review your data store configuration settings and then save them.

Steps

1. In the **Data Store Configuration** window, on the **Summary** tab, review your changes.

   Choose from:

   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.
2. Click **Done**.

Reviewing a local identity profile

Review or make changes to your local identity profile configuration.

About this task

Using the administrative console, on the **Summary** tab, amend, save, or discard your local identity profile changes.

Steps

1. In the **Data Store Configuration** window, on the **Summary** tab, review your changes.
   1. To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   2. To keep your changes, click **Done** and continue with the rest of the configuration.
   3. To discard your changes, click **Cancel**.

2. When finished, click **Done**.

Configuring the HTML Form Adapter for customer identities

After defining a local identity profile, associate it with an instance of the HTML Form Adapter for PingFederate to leverage the HTML Form Adapter to present users the options to authenticate through third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page.

About this task

Using the administrative console, on the **IdP Adapter** tab, create or add an HTML Form Adapter instance.

> **Note**
>
> For registration and profile management, ensure the HTML Form Adapter instance is configured to validate credentials stored in PingDirectory. This validation configuration is not required if your use case does not involve registration or profile management. For more information, see Enabling third-party identity providers without registration.

Steps

1. Go to the **Authentication > Integration > IdP Adapters**.
2. To create a new HTML Form Adapter instance, click **Create New Instance** or to reuse an existing instance, click on its name.
3. On the **IdP Adapter** tab, from the **Password Credential Validator Instance** list, select the LDAP Username Password Credential Validator instance that has been set up to validate credentials stored on your PingDirectory.

   > **Note**
   >
   > Skip this step if your use case does not involve registration or profile management.

4. On the **IdP Adapter** tab, from the **Local Identity Profile** list, select a local identity profile.
5. Click **Next** and complete the rest of the configuration tabs.
6. On the **Summary** tab, click **Done**. This will open the **Manager IdP Adapter Instances** window.
7. In the **Manager IdP Adapter Instances** window, click **Save** to save all changes.

Related links

- Configuring an HTML Form Adapter instance
- Setting up PingDirectory for customer identities
- Managing local identity profiles

Setting up self-service registration

PingFederate leverages the HTML Form Adapter to deliver a secure and easy-to-use customer authentication, registration, and profile management solution.

About this task

A typical self-service registration setup involves five components:

- A PingDirectory installation
- An authentication policy contract
- A local identity profile
- An HTML Form Adapter instance
- An IdP authentication policy

To illustrate the configuration steps, consider the following example.

You need to support a consumer registration use case, where users complete a self-service registration process to create their accounts and then access resources protected by multiple service providers. For a registration to complete successfully, a user must provide an email address, a first name, a last name, an optional mobile phone number, and a password. The email address is the user identifier. All attributes are sent to the service providers as per the partner agreements. You have already created a specific object class in the directory to store the user information. The object class name is `aPerson`, and the LDAP attributes are `mail`, `givenName`, `sn`, and `mobile`.

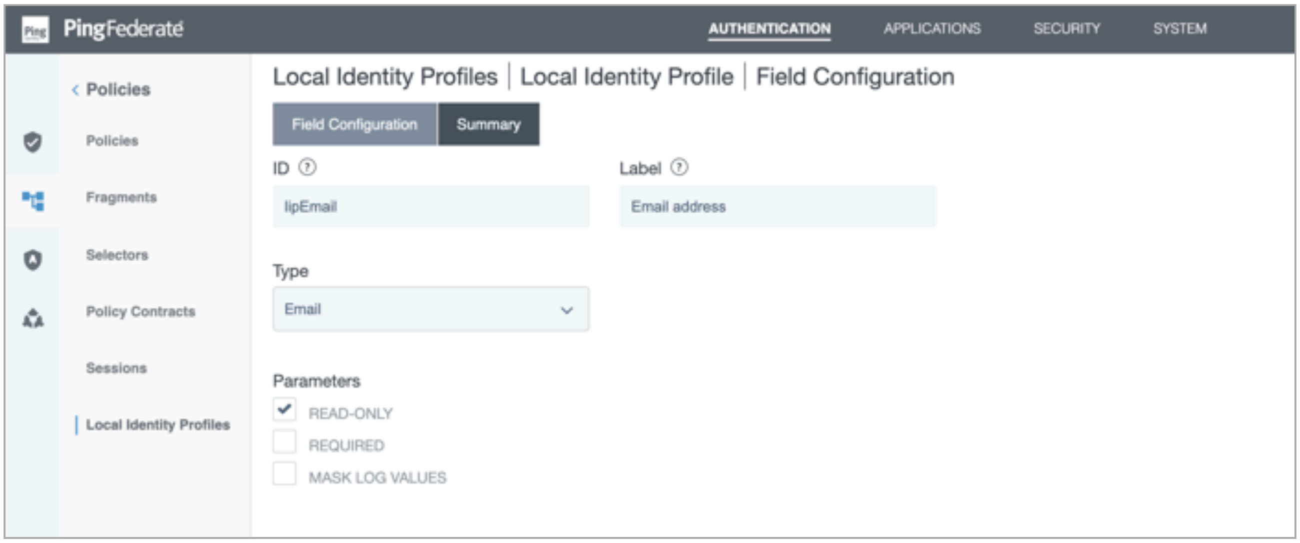
Steps

1. Install PingDirectory.
2. Create an authentication policy contract.
    1. Go to **Authentication** > **Policies** > **Policy Contracts**.
    2. In the **Policy Contracts** window, click **Create New Contract**.
    3. On the **Contract Attributes** tab, in the **Extend the Contract** field, extend the authentication policy contract with three additional attributes, such as, `firstName`, `lastName`, and `mobileNumber`.
    4. After each entry add, click **Add**. Click **Next**.
    5. On the **Summary** tab, review your changes. Click **Save**.

       For more information, see Managing policy contracts.
3. Create a local identity profile using the **Authentication** > **Policies** > **Local Identity Profiles** configuration wizard.
    1. In the **Local Identity Profiles** window, click **Create New Profile**.
    2. On the **Profile Info** tab, enter a name in the **Local Identity Profile Name** field.
    3. From the **Authentication Policy Contract** list, select the authentication policy, and select the **Enable Registration** check box. Click **Next**.
    4. On the **Authentication Sources** tab, click **Next**.
    5. On the **Fields** tab, click **Create New Field**.
    6. In the **Field Configuration** window, on the **Field Configuration** tab, define four local identity fields. Enter the information described in the following table.

Local Identity Profile fields and entries

Type	ID	Label	Parameters
Email	lipEmail	Email address	Select the **Required** check box.
Text	lipFirstName	First name	Select the **Required** check box.
Text	lipLastName	Last name	Select the **Required** check box.
Phone	lipMobile	Mobile number	No parameters are required.



Local Identity Profiles | Local Identity Profile | Field Configuration

ID: lipEmail

Label: Email address

Type: Email

Parameters

☑ READ-ONLY

□ REQUIRED

□ MASK LOG VALUES

7. After each field entry, click **Next**. On the **Summary** tab, review your changes. Click **Done**.
8. Repeat steps e through g until the fields are entered.

   As needed, select the **Mask Log Values** check box for any of the four local identity fields and the **Mask all OGNL-expression generated log values** check box. The latter applies to all local identity fields.
9. In the **Local Identity Profile** window, on the **Fields** tab, identify an ID field to be the unique ID for your configuration, and click the corresponding **Unique ID**. Click **Next**.
10. On the **Email Verification** tab, click **Next**.
11. On the **Registration** tab, click **Next**.
12. On the **Data Store Configuration** tab, click **Configure Data Store**.
13. In the **Data Store Configuration** window, on the **Data Store** tab, from the **Data Store** list, select the LDAP datastore that has been set up to connect to your PingDirectory. Click **Next**.
14. In the **Data Store Configuration** window, on the **LDAP Configuration** tab, specify the branch of your directory hierarchy where you want PingFederate to store customer identities in the **Base DN** field and the LDAP attributes to be associated with fields defined in this local identity profile under **Attribute**.
15. In the **Data Store Configuration** window, on the **Identity Creation** tab, define the RDN pattern in the **Relative DN Pattern** field, and select your object, such as class `aPerson` for this sample use case, from the **Object Class** list.

In the **Data Store Configuration** window, on the **Data Store Mapping** tab, configure the mapping between the local identity profile fields and the datastore attributes. See the following table.

Mapping entries for local identity profile fields and datastore attributes

Field	Data Store Attribute
lipEmail	mail
lipFirstName	givenName
lipLastName	sn
lipMobile	mobile

16. In the **Data Store Configuration** window, on the **Summary** tab, click **Done**.
17. On the **Summary** tab of the local identity profile, click **Save**.

    For more information, see Configuring local identity profiles.

    Configure an HTML Form Adapter instance for customer identities. Go to the **Authentication > Integration > IdP Adapters**.Create a new HTML Form Adapter instance or reuse an existing one by clicking its name.On the **IdP Adapter** tab, in the **Password Credential Validator Instance** section add the LDAP Username Password Credential Validator instance that has been set up to validate credentials stored on your PingDirectory.On the **IdP Adapter** tab, select the newly created local identity profile in the **Local Identity Profile** list.Complete the rest of the configuration and save all changes.

    (For more information, see Configuring the HTML Form Adapter for customer identities.)
18. Go to **Authentication > Policies > Policies**.
19. Click **Add Policy**.
20. In the **Policy** window, in the **Name** field, enter a name.
21. Select the HTML Form Adapter instance (configured in step 4) under **Policy**.
    1. For its **Fail** path, select **Done**.
    2. For its **Success** path, select the local identity profile (created in step 3).
22. Click **Local Identity Mapping** underneath the selected local identity profile, which opens the **Inbound Mapping & Contract Fulfillment** configuration wizard.

    At runtime, PingFederate fulfills the value of the `pf.local.identity.unique.id` built-in local identity field based on this configuration and passes the value to PingDirectory.
23. On the **Attribute Sources & User Lookup** tab, click **Next**.
24. On the **Contract Fulfillment** tab, fulfill the authentication policy contract with values from this local identity profile as follows:

Outbound Contract Fulfillment	Source	Value
subject	Local Identity	lipEmail
firstName	Local Identity	lipFirstName
lastName	Local Identity	lipLastName
mobileNumber	Local Identity	lipMobile

25. On the Issuance Criteria tab, click Next.

26. On the **Summary** tab, click **Done**.

27. On the **Policy** window, click **Done**.

28. On the **Policies** window, select the **IdP Authentication Policies** check box.

> **Note**
>
> Other IdP authentication policies, if any, are enabled as well.

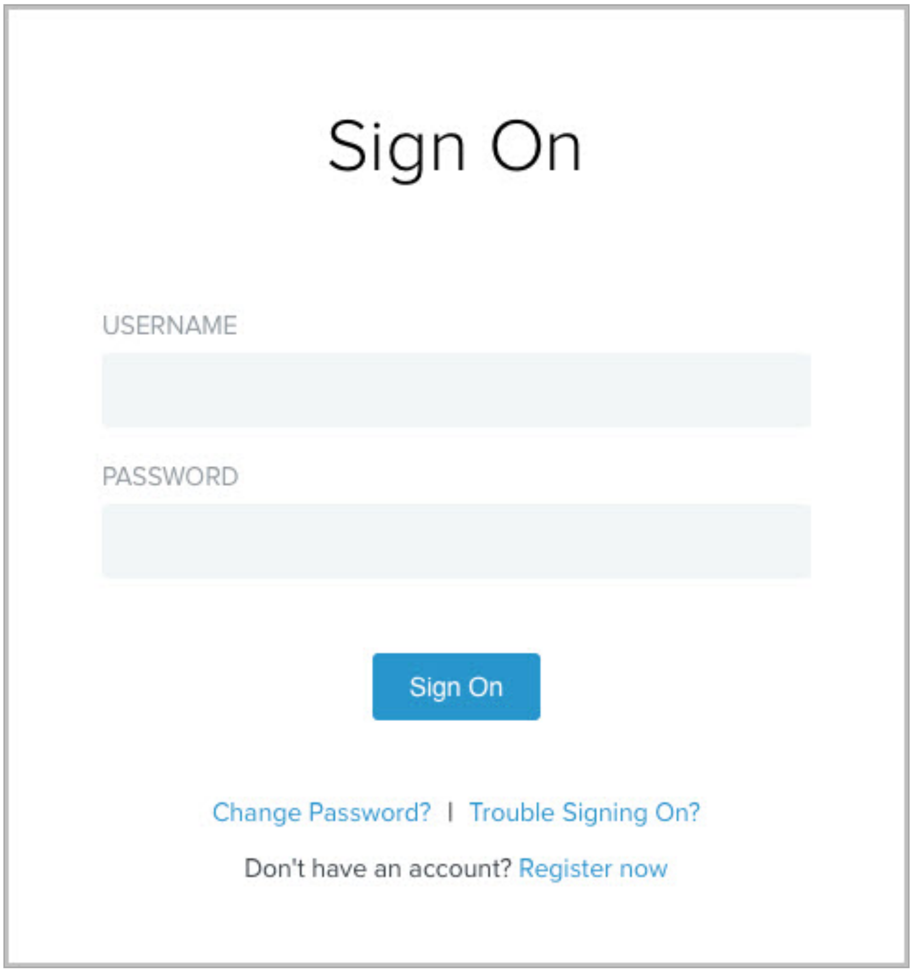
29. Click **Save** to keep your changes.

For more information, see Applying policy contracts or identity profiles to authentication policies.

4. Map the authentication policy contract to the applicable Browser SSO connections, OAuth grant-mapping configuration, or both. See Managing authentication source mappings and Managing authentication policy contract grant mapping.

Result

You have now successfully set up self-service registration. When users sign on through this HTML Form Adapter instance, they can complete a self-service registration process to create their accounts using the **Register now** link, as illustrated in the following screen capture.



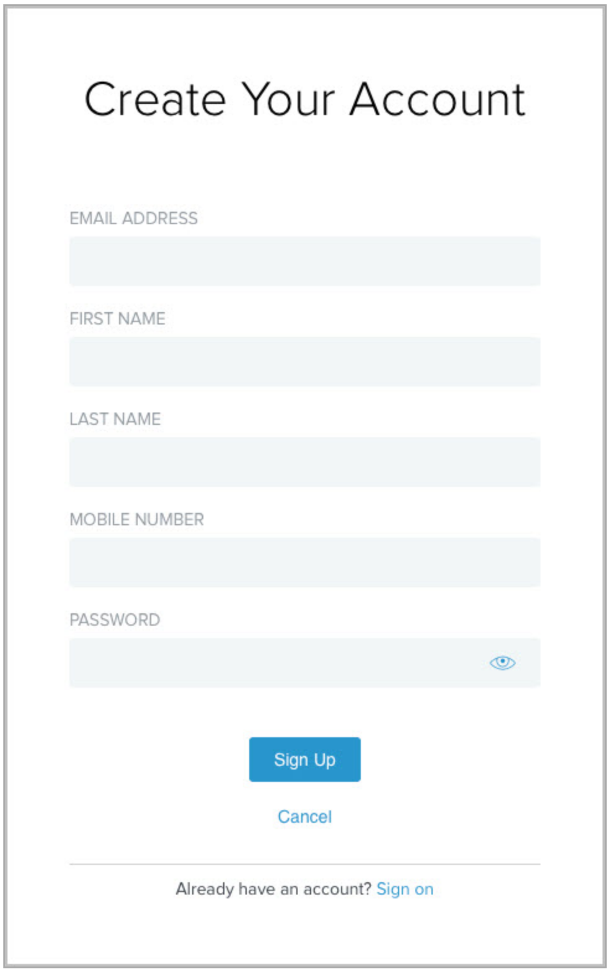
Sign On

USERNAME

PASSWORD

Sign On
Change Password? | Trouble Signing On?
Don't have an account? Register now

If a user chooses to register, the HTML Form Adapter redirects the user to the registration page. Based on the configuration of this sample use case, as illustrated in the following registration screen capture.



Create Your Account

EMAIL ADDRESS

FIRST NAME

LAST NAME

MOBILE NUMBER

PASSWORD

Already have an account? Sign on

Enabling third-party identity providers

For registration, you can optionally allow users to leverage their existing identities from third-party identity providers. Any identity provider (IdP) connection or IdP adapter, such as the LinkedIn IdP Adapter in the LinkedIn Login Integration Kit, can be used as an authentication source to a third-party identity provider.

About this task

Using the Administrative console, enable third-party IdP. This configuration involves the same five components to set up registration, plus the IdP connections or IdP adapter instances to connect with the third-party identity providers. This capability enables a mapping configuration between the attributes returned by the identity provider and the fields within the registration page, streamlining the registration process. See the following configuration steps.

- IdP connections or IdP adapter instances
- A PingDirectory installation (Install PingDirectory)
- An authentication policy contract ([step_ysz_y5y_vcb])
- A local identity profile ([step_iyh_11z_vcb])
- An HTML Form Adapter instance ([step_rnq_11z_vcb])
- An IdP authentication policy ([step_dm4_cdz_vcb])

To illustrate the configuration steps, consider the following example.

You need to support a consumer registration use case, where users complete a self-service registration process to create their accounts and then access resources protected by multiple service providers. For a registration to complete successfully, a user must provide an email address, a first name, a last name, an optional mobile phone number, and a password. The email address is the user identifier. All attributes are sent to the service providers as per the partner agreements. You have already created a specific object class in the directory to store the user information. The object class name is `aPerson`, and the LDAP attributes are `mail`, `givenName`, `sn`, and `mobile`.

Additionally, this use case must also allow users to take advantage of their existing accounts at ACME, a major social network, for registration and authentication. It happens that you have already established an IdP connection to this social network, from which you received the same set of attributes: `SAML_SUBJECT` for the user's email address, `ssoFirstName`, `ssoLastName`, and `ssoMobile`.

Configure a third-party identifier.

> **Tip**
>
> If you know how to set up PingDirectory to connect with PingFederate and an authentication policy contract shown in Setting up self-service registration, you can skip to [step_iyh_11z_vcb] to create a local identity profile.

Steps

1. Install PingDirectory.

   Create an authentication policy contract.Go to **Authentication** > **Policies** > **Policy Contracts**.In the **Policy Contracts** window, click **Create New Contract**.On the **Contract Attributes** tab, in the **Extend the Contract** field, extend the authentication policy contract with three additional attributes, such as, `firstName`, `lastName`, and `mobileNumber`.After each entry add, click **Add**. Click **Next**.On the **Summary** tab, review your changes. Click **Done**.In the **Policy Contracts** window, click **Save**.For more information, see Managing policy contracts.

2. Create a local identity profile using the **Authentication** > **Policies** > **Local Identity Profiles**. Click **Create New Profile**to access the **Local Identity Profile** window and configuration wizard.

   1. On the **Profile Info** tab, enter a name in the**Local Identity Profile Name** field, select the authentication policy from [step_ysz_y5y_vcb], and select the **Enable Registration** check box. Click **Next**.

   2. On the **Authentication Sources** tab, enter `ACME` in the **Authentication Source** field. Click **Add**.

      > **Note**
      >
      > To support additional third-party identity providers, enter a value for each. At runtime, the sign-on page displays them in the order defined on this window.

      > **Tip**
      >
      > If you know how to set up a local identity profile and an HTML Form Adapter instance for customer identities shown in Setting up self-service registration), you can skip to [step_dm4_cdz_vcb] to create an IdP authentication policy.

   3. On the **Fields** tab, define four local identity fields as shown in the following table.

Type	ID	Label	Parameters
Email	lipEmail	Email address	Select the **Required** and **Unique ID** check boxes. ⓘ **Note** The **Unique ID**check boxes are on the **Fields** tab.
Text	lipFirstName	First name	Select the **Required** check box.
Text	lipLastName	Last name	Select the **Required** check box.
Phone	lipMobile	Mobile number	None required.

As needed, select the **Mask Log Values** check box for any of the four local identity fields, and select the **Mask all OGNL-expression generated log values** check box for all fields.

4. On the **Email Verification** tab, click **Next**.

5. On the **Registration** tab, click **Next**.

6. On the **Data Store Configuration** tab, click **Configure Data Store**.

7. In the **Data Store Configuration** window, on the **Data Store** tab, from the **Data Store** list, select the LDAP datastore that has been set up to connect to your PingDirectory. Click **Next**.

8. In the **Data Store Configuration** window, on the **LDAP Configuration** tab, specify the branch of your directory hierarchy where you want PingFederate to store customer identities in the **Base DN** field and the LDAP attributes to be associated with fields defined in this local identity profile under **Attribute**.

9. In the **Data Store Configuration** window, on the **Identity Creation** tab, define the RDN pattern in the **Relative DN Pattern** field, and select your object, such as class `aPerson` for this sample use case, from the **Object Class** list.

   In the **Data Store Configuration** window, on the **Identity Creation** tab, define the RDN pattern in the **Relative DN Pattern** field, and select your object, such as class `aPerson` for this sample use case, from the **Object Class** list.

10. In the **Data Store Configuration** window, on the **Summary** tab, click **Done**.

11. On the **Summary** tab of the local identity profile, click **Save**.

    For more information, see Configuring local identity profiles.

    Configure an HTML Form Adapter instance for customer identities. Go to the **Authentication > Integration > IdP Adapters**.Create a new HTML Form Adapter instance or reuse an existing one by clicking its name.On the **IdP Adapter** tab, in the **Password Credential Validator Instance** section add the LDAP Username Password Credential Validator instance that has been set up to validate credentials stored on your PingDirectory.On the **IdP Adapter** tab, select the newly created local identity profile in the **Local Identity Profile** list.Complete the rest of the configuration and save all changes.

    (For more information, see Configuring the HTML Form Adapter for customer identities.)

3. Create an IdP authentication policy.

    1. Go to the **Authentication** > **Policies** > **Policies**.

    2. Under **Policy**, select the HTML Form Adapter instance configured in [step_rnq_11z_vcb].

    3. In the **Policy** window, under the **Value** section, select the drop-down arrow to show the **Fail** and **Success** fields.

        1. For the **Fail** path, select **Done**.

        2. For the **Success** path, select the local identity profile created in [step_iyh_11z_vcb]. Click **Done**.

    4. Click **Local Identity Mapping** underneath the selected local identity profile, which opens the **Inbound Mapping & Contract Fulfillment** configuration wizard.

        Note

        The next few steps configure the fulfillment of the authentication policy contract for the scenario where users choose to register directly without going through ACME.

        Tip

        If you know how to setup the inbound mapping and contract fulfillment of an authentication policy contract through a local identity profile shown in Setting up self-service registration, you can skip to [substep_ag2_wbg_wcb] to create a rule for the scenario where users choose to register and subsequently authenticate via ACME.

        On the **Inbound Mapping & Contract Fulfillment Inbound Mapping** window, configure the `pf.local.identity.unique.id` built-in local identity field for the registration process. At runtime, PingFederate fulfills the value of the `pf.local.identity.unique.id` built-in local identity field based on this configuration and passes the value to PingDirectory.

        PingDirectory uses this value to determine whether such identity has already been created. The `pf.local.identity.unique.id` field value should therefore be mapped from the subject identifier of the preceding authentication source, namely the `username` attribute from the HTML Form Adapter.

        For this sample use case, configure the **Inbound Mapping** window as shown in the following table.

Inbound Mapping Fulfillment	Source	Value
pf.local.identity.unique.id	Adapter	username

    5. On the **Attribute Sources & User Lookup** tab, click **Next**.

    6. On the **Attribute Sources & User Lookup** tab, click **Next**.

        On the **Issuance Criteria** tab, click **Next**.

    7. On the **Inbound Mapping & Contract Fulfillment** > **Summary** window, click **Done**.

        The **Inbound Mapping & Contract Fulfillment** configuration wizard brings back the **Manage Authentication Policies** window.

> **Note**
>
> The remaining steps configure the fulfillment of the authentication policy contract for the scenario where users choose to register and subsequently authenticate through ACME.

8. Click **Rules** underneath the **Success** path of the HTML Form Adapter instance.

9. On the **Rules** dialog, create a policy path for users who choose to register and authenticate via ACME. For this sample use case, configure as in the following table.

Authentication policy rules fields and entries

Attribute Name	Condition	Value	Result
policy.action	equal to	ACME **Important** The value here must match the value defined on the **Authentication Sources** window. See [substep_urn_4vy_vcb].	ACME users The **Result** field controls the label shown for the policy path of this rule. The value does not need to match the value defined on the **Authentication Sources** window.

> **Important**
>
> If you have defined multiple third-party identity providers on the **Authentication Sources** window, you must repeat these steps to add a `policy.action` rule to create a policy path for each.

In addition, select the **Default to Success** check box, the default behavior. When selected, the **Success** path remains, which is important for this sample use case where users are free to choose whether to register and subsequently authenticate via ACME.

When finished, click **Done**, which brings you back to the **Authentication Policies** window.

10. For the **ACME users** path, select the IdP connection to ACME under **Action**.

> **Tip**
>
> Generally speaking, any IdP adapter instance or IdP connection that connects to the third-party identity provider can be used here.

   1. For its **Fail** path, select **Done**.

> **Note**
>
> If you have defined multiple third-party identity providers and added rules to create a policy path for each, you may select **Restart**. The **Restart** policy action provides users the opportunity to do over. When triggered, the policy engine routes the requests back to the first checkpoint of the invoked authentication policy.
> By selecting **Restart** for the **Fail** path, you give users the opportunity to choose another third-party identity provider when they fail to authenticate through ACME.
> Undesirable looping behaviors can occur if you select **Restart** for the **Fail** path at the root of an authentication policy tree. PingFederate mitigates this risk by automatically limiting the number of policy restarts per transaction.

2. For its **Success** path, select the local identity profile created in [step_iyh_11z_vcb].

11. Click **Local Identity Mapping** underneath the selected IdP connection, which opens the **Inbound Mapping & Contract Fulfillment** configuration wizard.

12. On the **Inbound Mapping & Contract FulfillmentInbound Mapping** window, configure the `pf.local.identity.unique.id` built-in local identity field for the registration process and optionally other fields so that PingFederate can pre-populate values for these fields on the registration page.

    At runtime, PingFederate fulfills the value of the `pf.local.identity.unique.id` built-in local identity field based on this configuration and passes the value to PingDirectory.

    PingDirectory uses this value to determine whether such identity has already been created. The `pf.local.identity.unique.id` field value should therefore be mapped from the subject identifier of the preceding authentication source

    For this sample use case, the **Inbound Mapping** window is configured as in the following table.

    Inbound Mapping fields and entries

Inbound Mapping Fulfillment	Source	Value
pf.local.identity.unique.id	IdP Connection	SAML_SUBJECT
lipEmail	IdP Connection	SAML_SUBJECT
lipFirstName	IdP Connection	ssoFirstName
lipLastName	IdP Connection	ssoLastName
lipMobile	IdP Connection	ssoMobile

13. On the **Attribute Sources & User Lookup** tab, click **Next**.

14. On the **Attribute Sources & User Lookup** tab, click **Next**.

4. On the **Issuance Criteria** tab, click **Next**.

    1. On the **Inbound Mapping & Contract Fulfillment Summary** tab, click **Done**.

The **Inbound Mapping & Contract Fulfillment** configuration wizard brings back the **Manage Authentication Policies** window.

> **Important**
>
> If you have defined multiple rules, each forming a policy path for a third-party identity provider, ensure you complete the **Inbound Mapping & Contract Fulfillment** configuration for each of them.

2. In the **Policies** window, on the **Policies** tab, select the **IdP Authentication Policies** check box.

> **Note**
>
> Other IdP authentication policies, if any, are enabled as well.

3. Click **Save** to keep your changes.

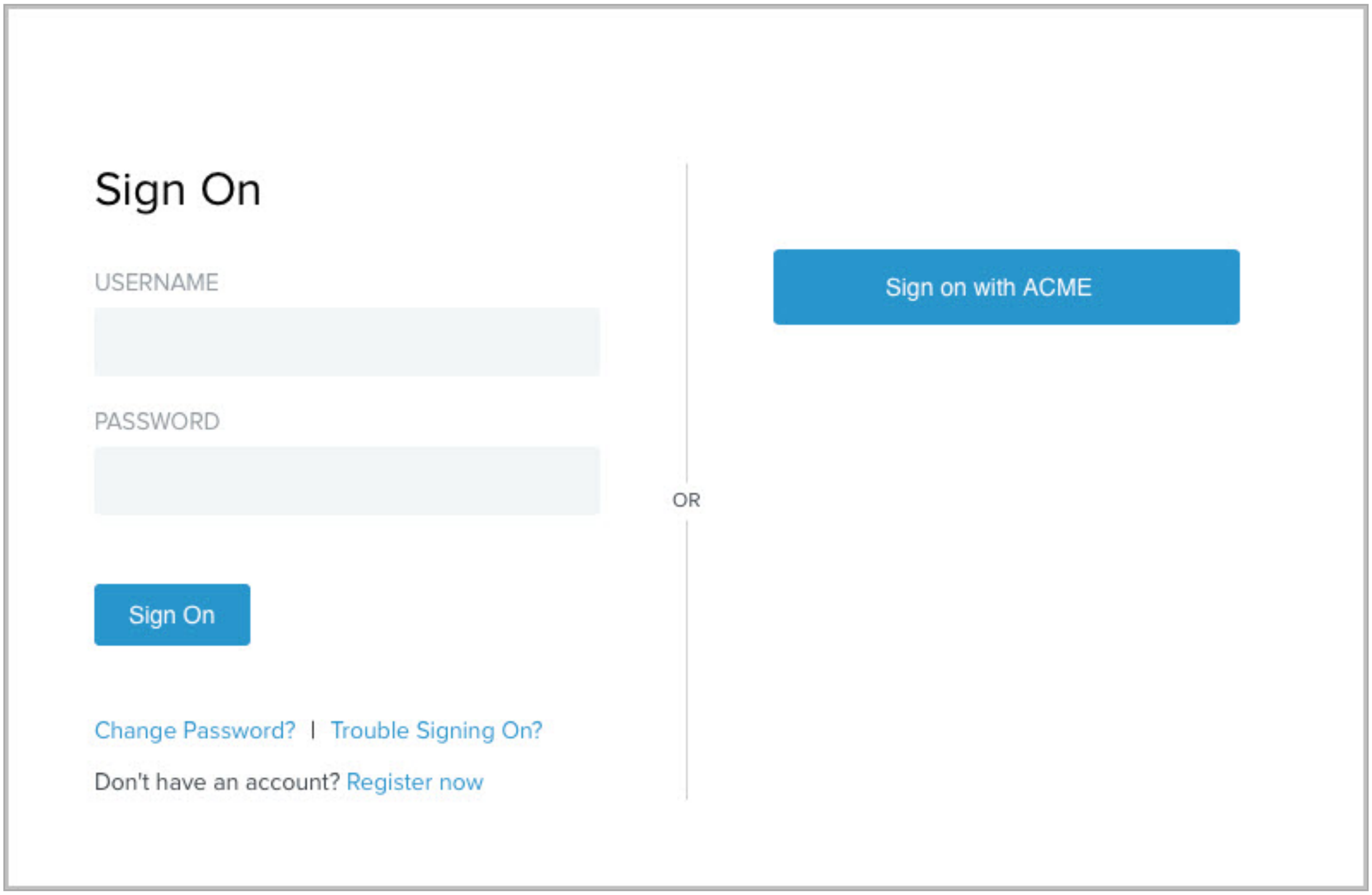
For more information, see Applying policy contracts or identity profiles to authentication policies.

Result

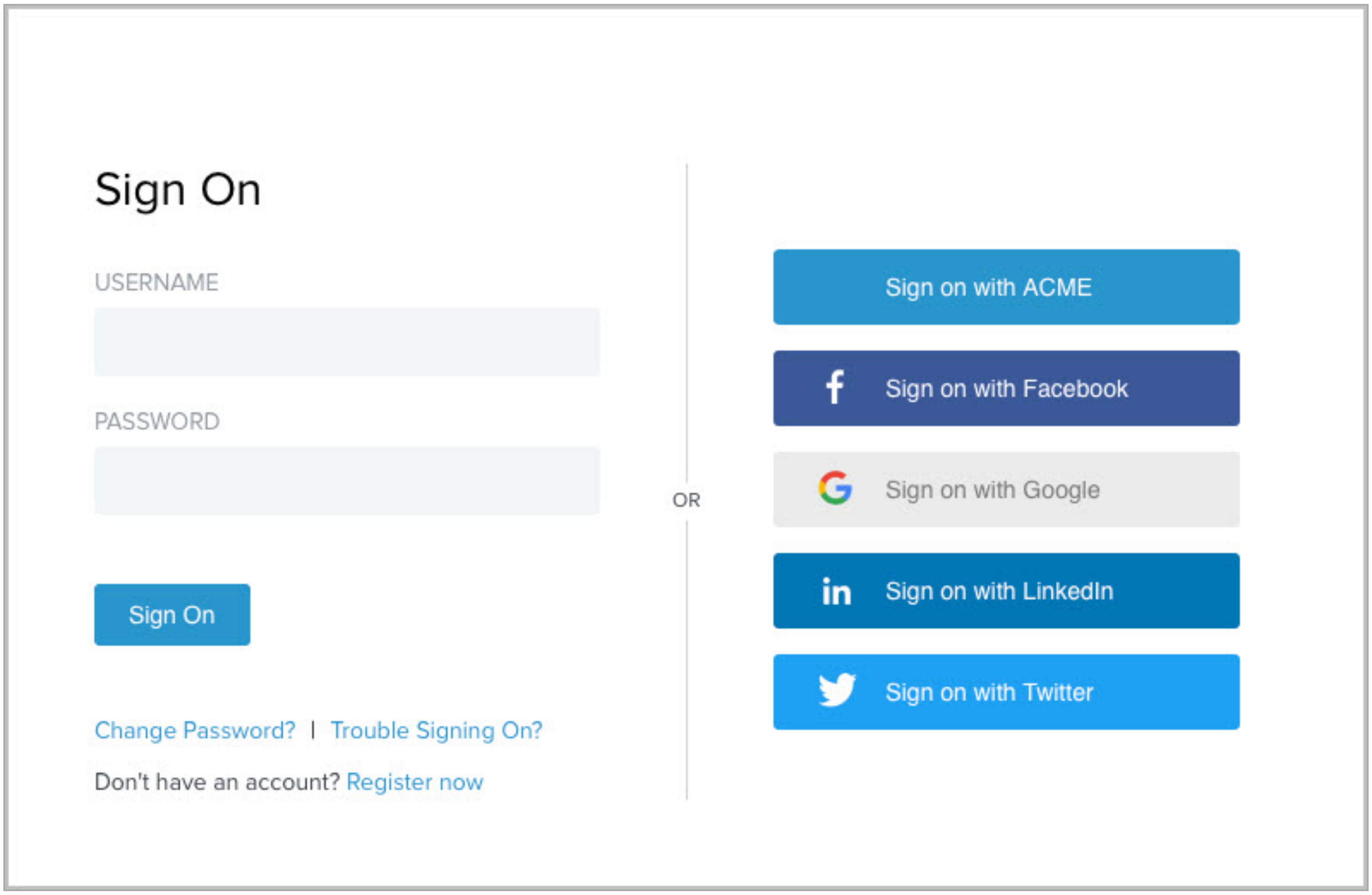
You have now successfully set up self-service registration with an option for users to register and subsequently authenticate via ACME. When users sign on through this HTML Form Adapter instance, they have two registration options:

- Click the **Register now** link, fill in the registration page, and register.
- Click the social sign-on link, authenticate via ACME, review the registration page, and register.

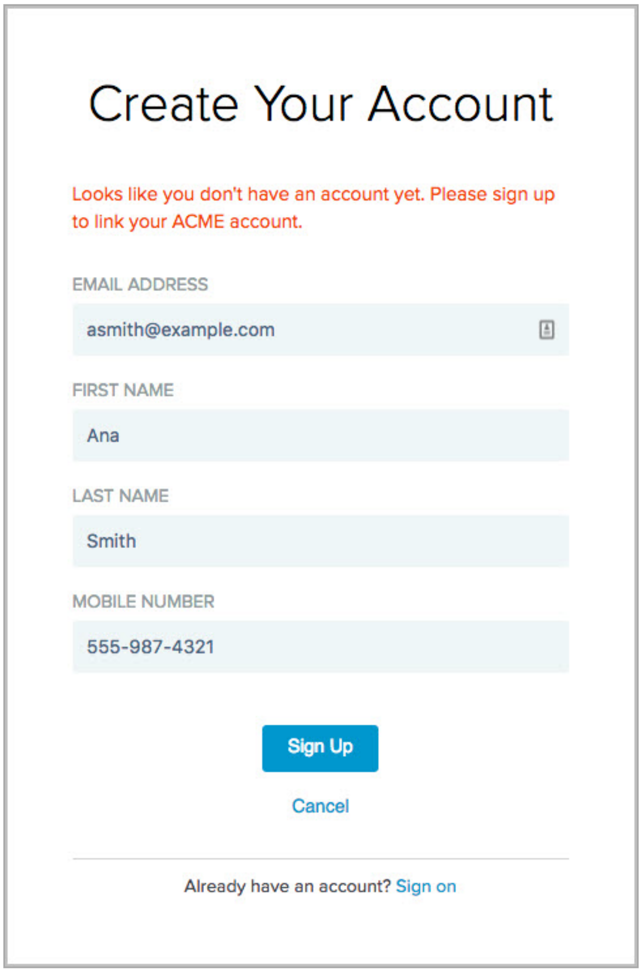
Based on the configuration of this sample use case, the following screen capture of a sign-on page is presented.


Sign On
USERNAME
PASSWORD
Sign On
Change Password? | Trouble Signing On?
Don't have an account? Register now
OR
Sign on with ACME

If you have added Facebook, Google, LinkedIn, and Twitter as the authentication sources, the following sign-on page is presented.



Suppose a user chooses to register through ACME. Once authenticated and redirected back to PingFederate, PingFederate pre-populates the registration page with values it receives from ACME, as illustrated in this screen capture.



Create Your Account

Looks like you don't have an account yet. Please sign up to link your ACME account.

EMAIL ADDRESS: asmith@example.com

FIRST NAME: Ana

LAST NAME: Smith

MOBILE NUMBER: 555-987-4321

This registration option streamlines the self-service registration process.

Related links

- Troubleshooting registration and profile management issues

Enabling profile management

In addition to registration, you can enable self-service profile management and specify which local identity fields users can update on the profile management page.

About this task

Use the administrative console to enable profile management.

To illustrate the configuration steps, consider the sample use case in Setting up self-service registration or Enabling third-party identity providers with the added requirement of allowing users to modify their mobile number and to remove their local accounts.

Configuration steps:

> **Tip**
>
> As the required components remain the same, the step sequence matches those in Setting up self-service registration and Enabling third-party identity providers as well. If you require more information for a given step, see the same step in one of the aforementioned pages.

Steps

1. Set up PingDirectory to connect with PingFederate.
2. Create an authentication policy contract. For more information on how to create an authentication policy contract, see Managing policy contracts.
3. Configure profile management when creating a new or reconfiguring an existing local identity profile.
    1. Go to **Authentication** > **Policies** > **Local Identity Profiles** configuration wizard.
    2. On the **Profile Info** tab, select the **Enable Profile Management** check box. Click **Next**.
    3. **Optional:** On the **Authentication Sources** tab, define authentication sources. For more information, see Defining authentication sources.
    4. On the **Fields** tab, select the **Profile Management** check box under **Applies To** for the applicable fields as you define local identity fields.

        These selected local identity fields will be shown to authenticated users on the profile management page.

    For this sample use case, select the **Profile Management** check box for the `lipMobile` local identity field.

    +

    1. On the **Email Verification** tab, click **Next**.
    2. On the **Registration** tab, click **Next**.
    3. On the **Profile Management** tab, select the **Enable Profile Deletion** check box.

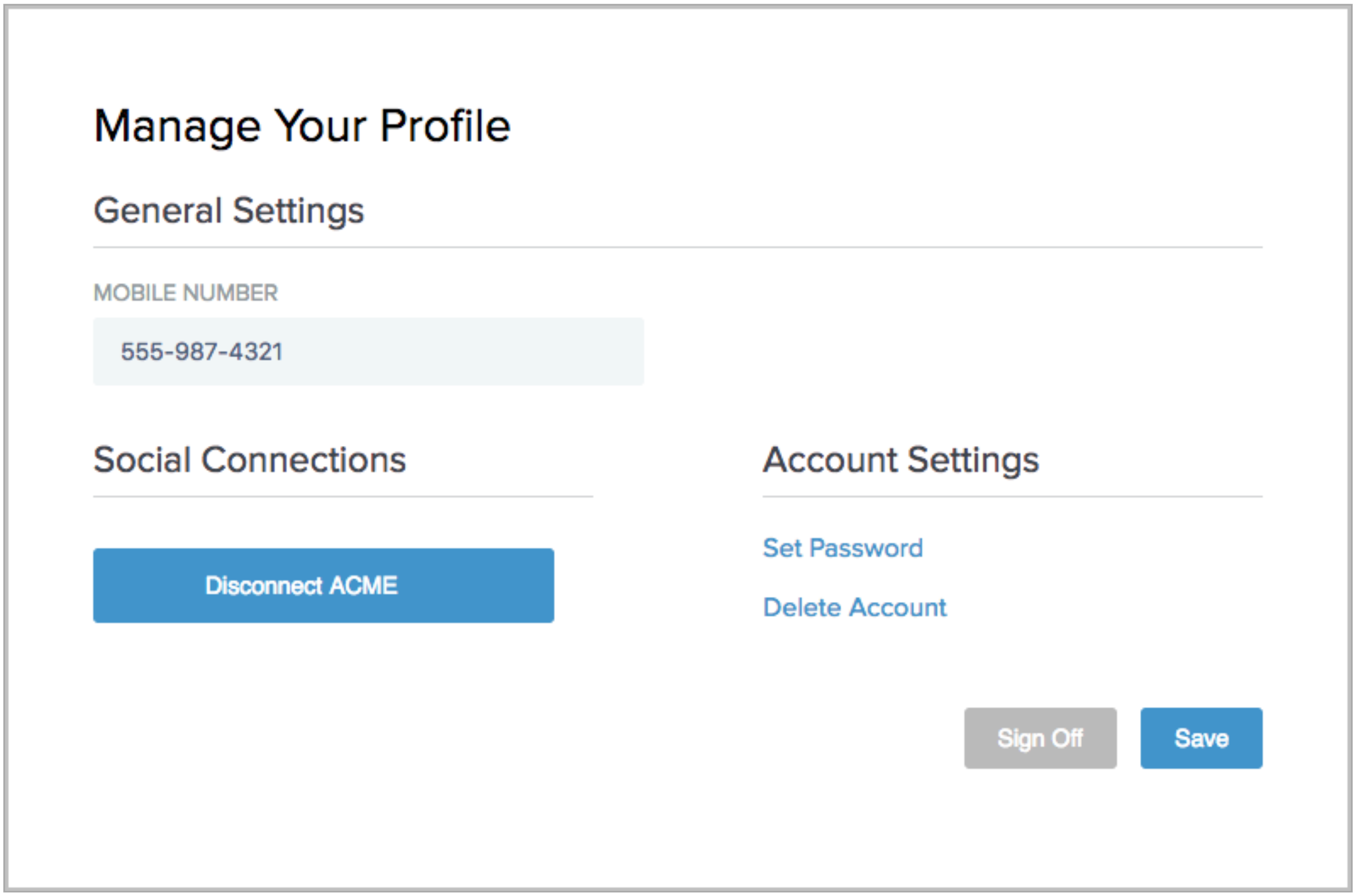
        In general, this is an optional feature. It is selected here because it is one of the requirements of this sample use case.

    4. Continue from step 3f as documented in Setting up self-service registration or Enabling third-party identity providers.
4. Configure an HTML Form Adapter instance for customer identities. For more information, see Configuring the HTML Form Adapter for customer identities.
5. Create an IdP authentication policy. For more information, see Defining authentication policies.
6. Provide the profile management URL to users.
    1. Go to **Authentication** > **Policies** > **Local Identity Profiles**.
    2. Select the local identity profile that you have configured profile management in [step_o3x_zvl_wcb].
    3. Copy the profile management URL as shown on the **Summary** tab and pass it to the users.

Result

You have now successfully enabled profile management. Authenticated users can review and modify the local identity fields that have been configured to show on the profile management page and delete their local accounts if the option has been enabled.

The following screen capture provides a sample of the profile management page based on the sample use case.



Manage Your Profile

General Settings

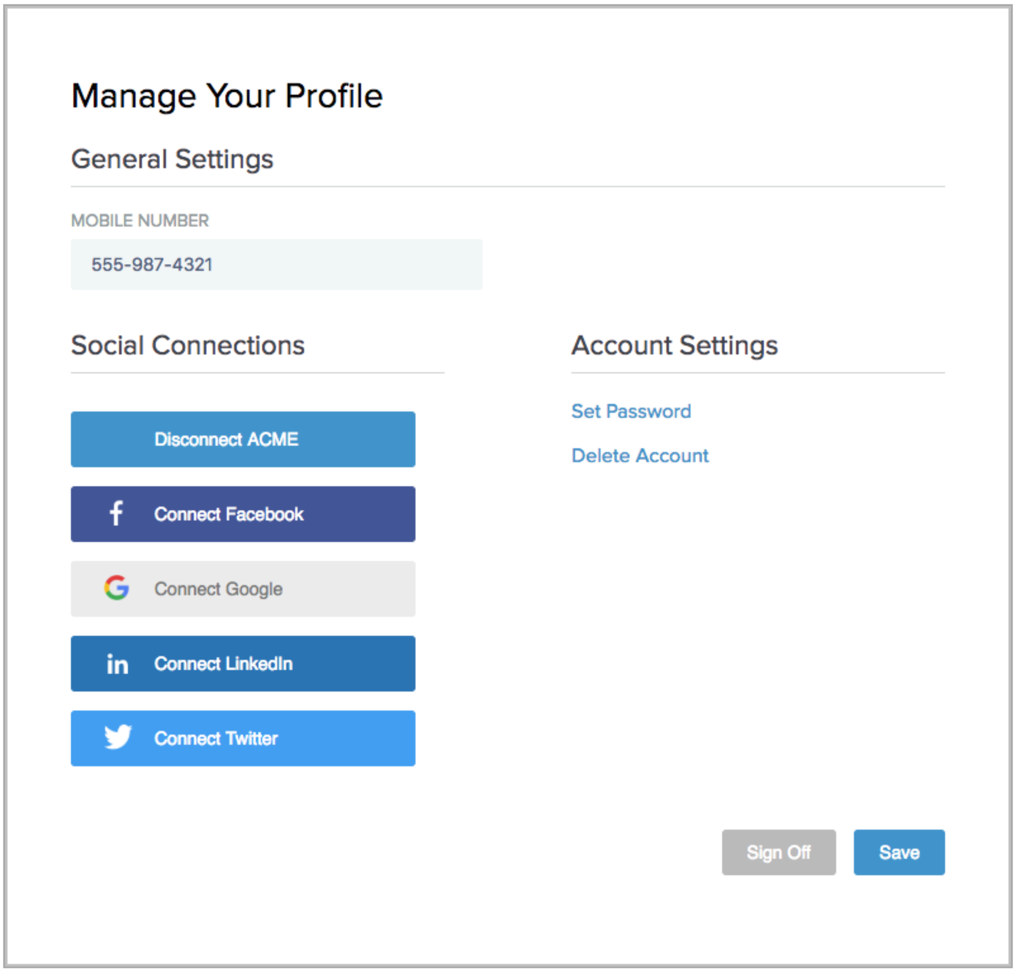
MOBILE NUMBER

555-987-4321

Social Connections

Account Settings

If you add Facebook, Google, LinkedIn, and Twitter to the local identity profile, when a user accesses the profile management page, the user will see a page similar to the following screen capture.



Manage Your Profile

General Settings

MOBILE NUMBER: 555-987-4321

Social Connections

Account Settings

If you have only one authentication source, the profile management page reminds the users that they must set a password for their local accounts before disconnecting the third-party identity provider.

Related links

- IdP endpoints

Creating advanced registration mapping

PingFederate leverages the HTML Form Adapter to deliver a secure and easy-to-use customer authentication, registration, and profile management solution. The HTML Form Adapter contract includes two core attributes.

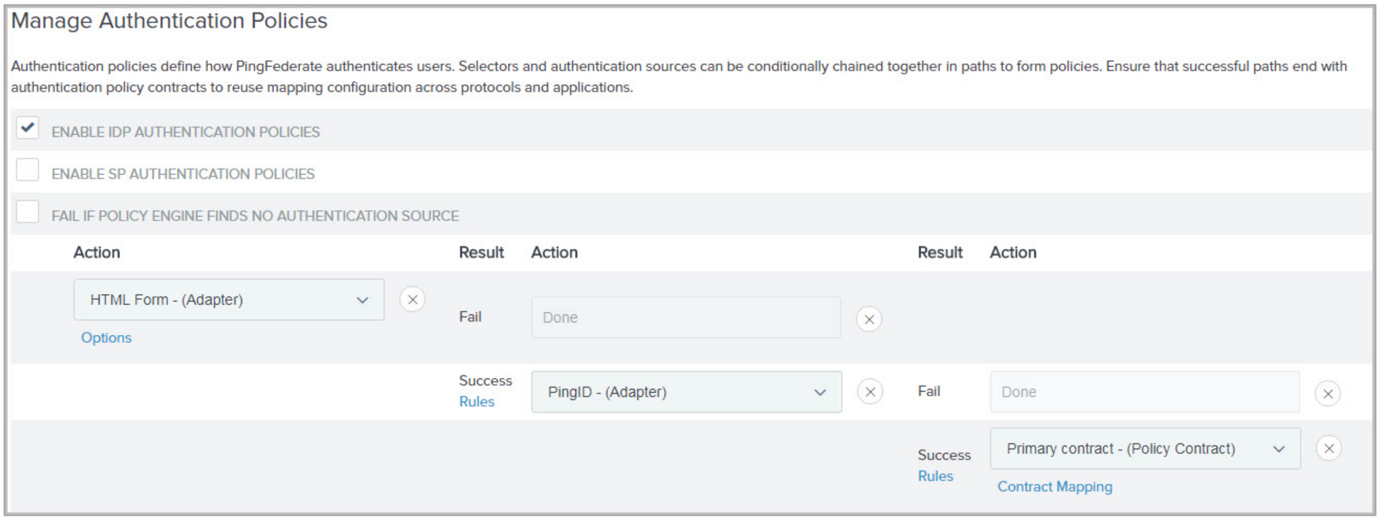
About this task

Using the administrative console, create advanced registration mapping by defining authentication policies in your adapter instances.

To illustrate the configuration steps, consider the following setup that you have already made with the parameters `username` and `policy.action`.

Whether or not the local identity profile is configured with any authentication sources, if the user chooses to register directly by clicking on the **Register now** link, PingFederate sets the value to `identity.registration`. This fulfillment allows you to create rules to differentiate authentication requirements from the registration flow

- A PingDirectory installation with a set of users.
- An LDAP datastore, an LDAP Username Password Credential Validator instance, and an HTML Form Adapter instance on PingFederate to validate credentials stored in PingDirectory.
- An IdP authentication policy that chains the HTML Form Adapter instance, an PingID Adapter instance, and an authentication policy contract for the purpose of enforcing PingID multi-factor authentication in multiple browser-based single sign-on (SSO) use cases through service provider (SP) connections, OAuth authorization code flow, and OAuth implicit flow. The following window capture illustrates your existing policy.



Manage Authentication Policies

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☑ ENABLE IDP AUTHENTICATION POLICIES

□ ENABLE SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Action	Result	Action	Result	Action
HTML Form - (Adapter) Options	Fail	Done		
	Success Rules	PingID - (Adapter)	Fail	Done
			Success Rules	Primary contract - (Policy Contract) Contract Mapping

To illustrate the configuration steps, consider the following setup that you have. You need to add support for a consumer registration use case similar to the one in Setting up self-service registration, and, at the same time, keep the policy that enforces the multi-factor authentication requirement.

Configuration steps.

Steps

1. Set up PingDirectory for customer identities.
2. Make a note of which authentication policy contract is currently being used in your policy.
3. Create a local identity profile.

    1. G to **Authentication** > **Policies** > **Local Identity Profiles** configuration wizard.
    2. On the **Profile Info** tab, in the **Local Identity Profile Name** field, enter a name of the local identity profile, and from the **Authentication Policy Contract** list, select the authentication policy from step 2. Select the **Enable Registration** check box.

        If you want to enable profile management as well, select the relevant check box.

4. Configure the HTML Form Adapter instance for customer identities.
    1. Go to **Authentication** > **Integration** > **IdP Adapters**.
    2. From the **Instance Name** section, select the **HTMLFormAdapter**.
5. Modify your existing IdP authentication policy.
    1. Go to **Authentication** > **Policies** > **Policies**.
    2. On the **Policies** tab, under the **Policy** section, click the relevant policy to open the **Policy** window.
    3. Under the **Success** path of the HTML Form Adapter instance, click **Rules**.
    4. In the **Rules** dialog, create a policy path for users who choose to register. For this sample use case, configure as follows.

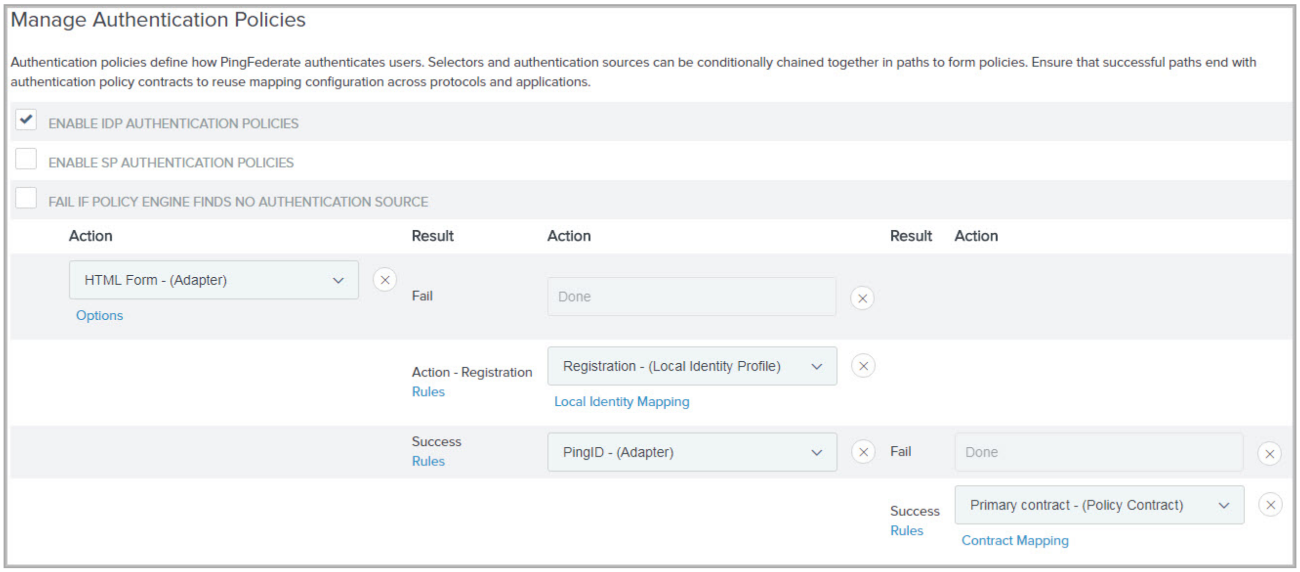
       Defining authentication policy rules attributes field and values

Attribute Name	Condition	Value	Result
policy.action	equal to	identity.registration	Registration Complete the rest of the configuration to create the local identity. The **Result** field controls the label shown for the policy path of this rule.

       In addition, ensure the **Default to Success** check box is selected. When selected, the **Success** path remains, which is important for this sample use case where users are redirected to the PingID Adapter instance to fulfill the multi-factor authentication requirement after authenticating successfully against the HTML Form Adapter.

       When finished, click **Done**, which brings you back to the **Manage Authentication Policies** window.
    5. For the **Registration** path, select the local identity profile from step 3 under **Action** and then complete its **Local Identity Mapping**configuration.

       The following screen capture illustrates your new policy.



Manage Authentication Policies

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☑ ENABLE IDP AUTHENTICATION POLICIES

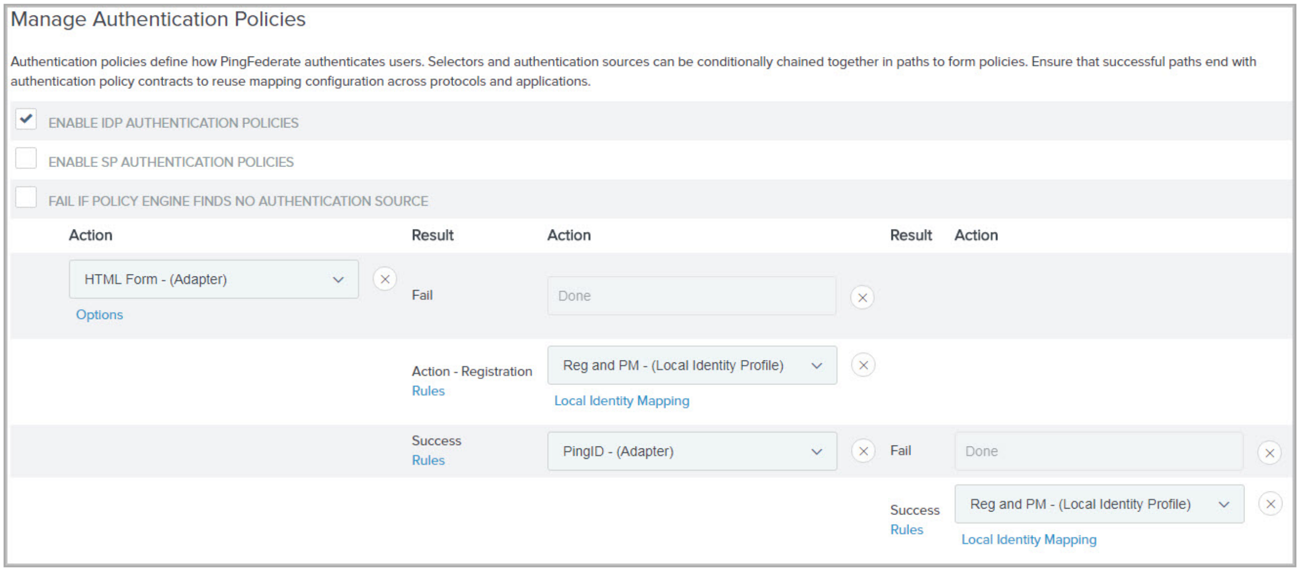
□ ENABLE SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

6. If you have enabled profile management in step 3, you must also replace the policy contract with the local identity profile and then complete its **Local Identity Mapping** configuration.

   This step is required so that PingFederate can route users through the **HTML Form > PingID** policy path when they try to access the profile management page.

   The following screen capture illustrates this change.



Manage Authentication Policies

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☑ ENABLE IDP AUTHENTICATION POLICIES

□ ENABLE SP AUTHENTICATION POLICIES

□ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

> **Note**
>
> No reconfiguration is required in your Browser SSO connections and OAuth grant-mapping configuration for your new policy to take effect.

7. Click **Save** to keep your changes.

Result

You have now successfully added the requested consumer registration and profile management use case to your current policy.

Related links

- Setting up PingDirectory for customer identities
- Managing local identity profiles
- Applying policy contracts or identity profiles to authentication policies

Enabling third-party identity providers without registration

If you have already configured identity provider (IdP) connections or IdP adapters to connect with third-party identity providers, you can enhance the HTML Form Adapter sign-on page with the option to authenticate with these providers.

About this task

You configure an IdP authentication policy to chain the HTML Form Adapter instance and an authentication policy contract. Then the policy contract can harness attribute values returned by the HTML Form Adapter instance for multiple browser-based single sign-on (SSO) use cases through service provider (SP) connections, OAuth authorization code flow, and OAuth implicit flow.

The following procedure offers an example of how you could enhance the sign-on experience by giving users the option to authenticate with their local accounts or their existing accounts on a major social network to which you have already established an IdP connection. In this example, the social network is named "ACME".

> **Tip**
>
> You can also deploy and configure Cloud Identity Connectors to support identities from Facebook, Google, LinkedIn, or Twitter.

Steps

1. Verify that the IdP connection to ACME returns the attributes required to complete the browser-based SSO use cases.
2. Note which authentication policy contract your policy uses.
3. Create a local identity profile:
   1. Go to **Authentication** > **Policies** > **Local Identity Profiles**. Click **Create New Profile**.
   2. On the **Profile Info** tab, in the **Local Identity Profile Name** field, enter a name.
   3. From the **Authentication Policy Contract**, select the authentication policy contract from step 2. Click **Next**.
   4. On the **Authentication Sources** tab, under **Authentication Source**, enter `ACME`, and then click **Add**. Click **Done**.

   > **Note**
   >
   > To support additional third-party identity providers, enter a value for each. At runtime, the sign-on page displays them in the order defined on this window.
4. Configure the HTML Form Adapter instance for customer identities:
   1. Go to **Integration** > **IdP Adapters**.
   2. On the **IdP Adapters** window, from the **Instance Name** list, click the **HTMLFormAdapter** instance.

5. Modify your existing IdP authentication policy:

    1. Go to **Authentication** > **Policies** > **Policies**.

    2. On the **Policies** tab, in the **Policy** section, click the existing IdP policy.

    3. In the **Success** path of the HTML Form Adapter instance, click **Rules**.

    4. In the **Rules** dialog, create a policy path for users who choose to authenticate with ACME. For this example, configure the fields as shown in the following table.

Attribute Name	Condition	Value	Result
policy.action	equal to	ACME **Important** The value here must match the value defined on the **Authentication Sources** tab. See [substep_afh_c4b_xcb].	ACME users **Note** The **Result** field controls the label shown for the policy path of this rule. The value does not need to match the value defined on the **Authentication Sources** window.

> **Important**
>
> If you have defined multiple third-party identity providers on the **Authentication Sources** tab, you must repeat these steps to add a `policy.action` rule to create a policy path for each.

    In addition, ensure the **Default to Success** check box is selected. When selected, the **Success** path remains, which is important when users can also authenticate using their local accounts, like in this example.

    5. Click **Done**.

    Result:

    This returns you to the **Authentication Policies** window.

    6. For the **ACME users** path, under **Action**, select the IdP connection to ACME.

> **Tip**
>
> Generally, any IdP adapter instance or IdP connection that connects to the third-party identity provider can be used here.

    7. For the ACME **Fail** path, select **Done**.

> **Note**
>
> If you have defined multiple third-party identity providers and added rules to create a policy path for each, you can select **Restart**. The **Restart** policy action provides users the opportunity to do over. When triggered, the policy engine routes the requests back to the first checkpoint of the invoked authentication policy. By selecting **Restart** for the **Fail** path, you give users the opportunity to choose another third-party identity provider if they fail to authenticate through ACME.

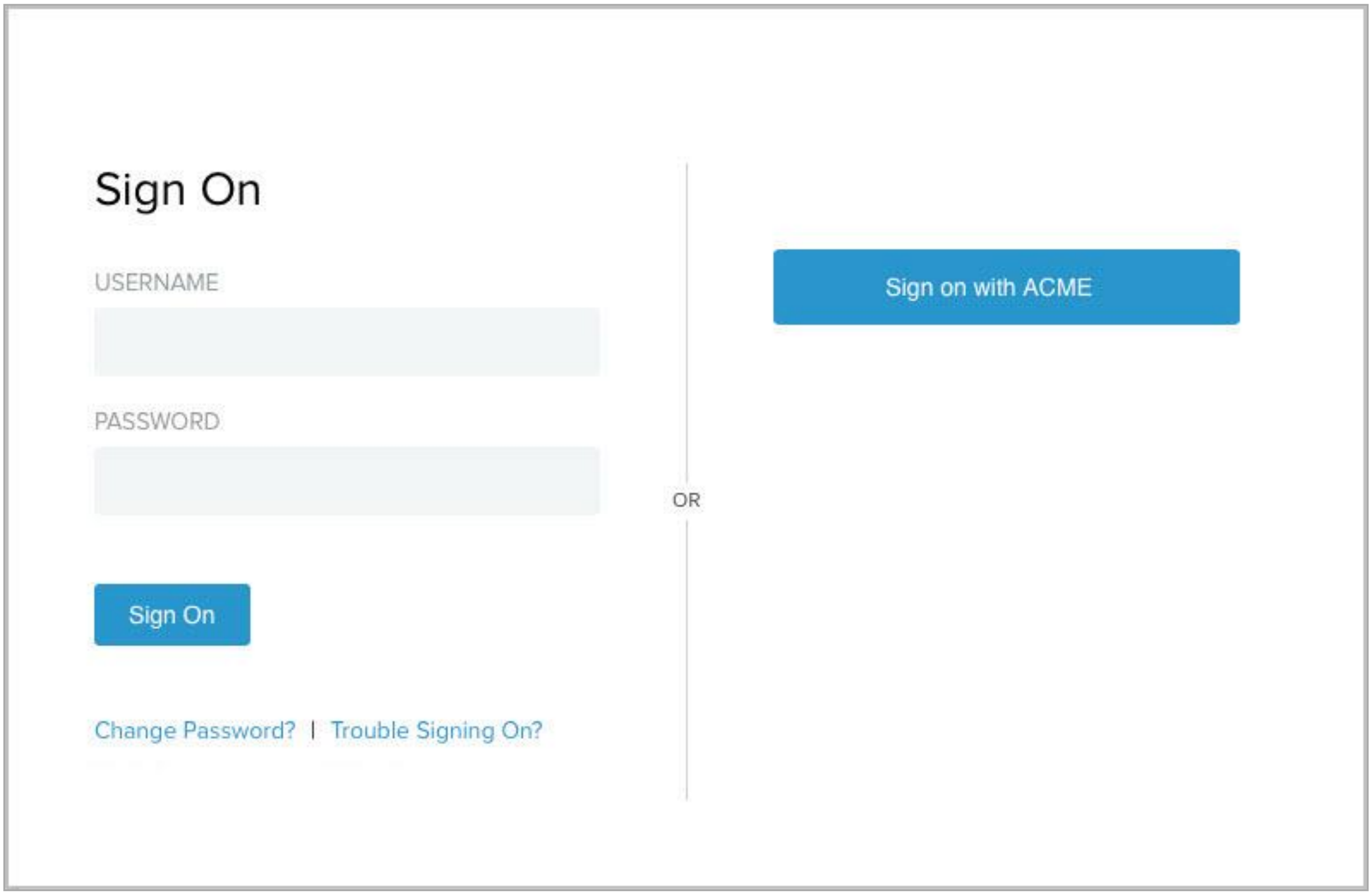
8. For the ACME **Success** path, select the local identity profile created in step 3, and then complete the **Local Identity Mapping** configuration.

> **Note**
>
> Because this use case does not involve registration, the source of fulfillment is limited to the preceding IdP connection or IdP adapter instance, dynamic text, and attribute mapping expression, if enabled.

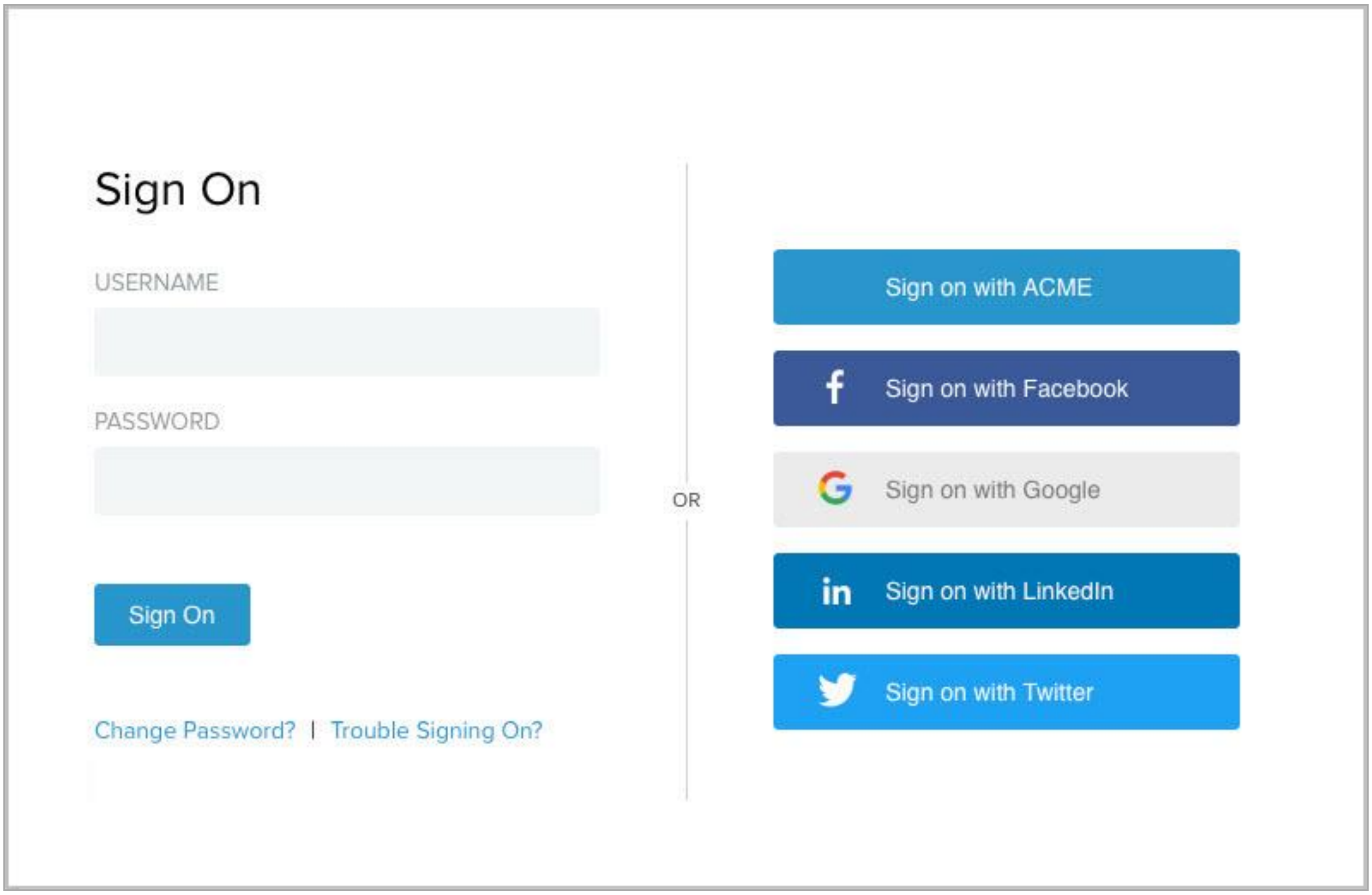
9. Click **Save**.

Result

After you give users the option to authenticate with ACME without enabling registration, when users sign on through this HTML Form Adapter instance, the following sign-on page is presented.



If you also added Facebook, Google, LinkedIn, and Twitter as authentication sources, the following sign-on page is presented.



Sign On

USERNAME

PASSWORD

Customizing assertions and authentication requests

Customize applicable messages by enabling OGNL expression and going to the **URL** window to access the **Show Advanced Customizations** option.

About this task

Some browser single sign-on (SSO) use cases might require additional customizations in the assertions sent from the PingFederate identity provider (IdP) server to the service provider (SP), or in the authentication requests sent from the PingFederate SP server to the IdP. PingFederate can fulfill these use cases on a per-connection basis using OGNL expressions.

Steps

1. Enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
2. Select the applicable SP or IdP connection.
3. On the **Activation & Summary** window, scroll to the Protocol Settings section, and click **Assertion Consumer Service URL** for an SP connection, or click **SSO Service URLs** for an IdP connection.
4. Click **Show Advanced Customizations** to customize the applicable message.

The available customizable **Message Types** vary depending on your federation role (IdP or SP) as well as the protocol of the connection (SAML 1.x, SAML 2.0, and WS-Federation). After you select a message type, you have access to its list of **Available Variables**. You can customize the assertions or the authentication requests as needed.

Related links

- Attribute mapping expressions
- Enabling and disabling expressions

Message types and available variables

Advanced customizations depend on available message types and available variables for both service provider (SP) connections and identity provider (IdP) connections.

The following tables describe the relationship between message type and available variable, as well as the corresponding class or interface information in Javadoc.

> **Tip**
>
> The Javadoc for PingFederate is located in the `<pf_install>/pingfederate/sdk/doc` directory.

SP connections (SAML 2.0)

Message Types	Available VariablesClasses/Interfaces in Javadoc

`AssertionType`	`#AssertionType`
	`org.sourceid.saml20.xmlbinding.assertion.AssertionType`
	`#AssertionTypes`
	`org.sourceid.saml20.xmlbinding.assertion.AssertionType[]`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

`ResponseDocument`	`#ResponseDocument`
	`org.sourceid.saml20.xmlbinding.protocol.ResponseDocument`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

SP connections (SAML 1.x)

Message Types	Available VariablesClasses/Interfaces in Javadoc

`AssertionType`	`#AssertionType`
	`org.sourceid.protocol.saml11.xml.AssertionType`
	`#AssertionTypes`
	`org.sourceid.protocol.saml11.xml.AssertionType[]`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

Message Types	Available VariablesClasses/Interfaces in Javadoc

`ResponseDocument`	`#ResponseDocument`
	`org.sourceid.protocol.samlp11.xml.ResponseDocument`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

SP connections (WS-Federation)

Message Types	Available VariablesClasses/Interfaces in Javadoc

`AssertionType`	`#AssertionType`
	`org.sourceid.protocol.saml11.xml.AssertionType`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

`RequestSecurityToken ResponseDocument`	`#RequestSecurityTokenResponseDocument`
	`org.xmlsoap.schemas.ws.x2005.x02.trust.RequestSecurityTokenResponseDocument`
	`#Attributes`
	`org.sourceid.util.log.AttributeMap`

IdP connections (SAML 2.0)

Message Type	Available VariablesClasses/Interfaces in Javadoc
`AuthnRequestDocument`	`#AuthnRequestDocument` `org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument`

Other available variables (regardless of roles and protocols)

Available Variables	Classes/Interfaces in Javadoc
`#XmlHelper`	`com.pingidentity.sdk.xml.XmlHelper`
`#HttpServletRequest`	`javax.servlet.http.HttpServletRequest`
`#HttpServletResponse`	`javax.servlet.http.HttpServletResponse`

Variables related to Federation Hub (regardless of message type)

Connections	Protocol	Available VariablesClasses/Interfaces in Javadoc
SP and IdP connections	SAML 2.0	`#FedHubIncomingAuthnRequest` `org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument`

Connections	Protocol	Available VariablesClasses/Interfaces in Javadoc
SP connection	SAML 2.0	`#FedHubOutgoingAuthnRequest` `org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument`
SP connection	SAML 2.0 SAML 1.x WS-Federation	`#FedHubIncomingAuthnResponse` `org.sourceid.saml20.xmlbinding.protocol.ResponseDocument` (SAML 20) `org.sourceid.protocol.samlp11.xml.ResponseDocument` (SAML 1.x) `org.xmlsoap.schemas.ws.x2005.x02.trust.RequestSecurityTokenResponseDocument` (WS-Federation)

SP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubIdpConnPartnerId`
		`java.lang.String`
		The **Partner's Entity ID** in the IdP connection that bridges the identity provider.

SP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubIdpConnProtocol`
		`java.lang.String`
		The protocol of the SP connection. The returned values are `SAML20`, `SAML11`, `SAML10`, or `WSFED`.

IdP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubSpConnApplicationName`
		`java.lang.String`
		The application name in the SP connection that bridges the service provider.

IdP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubSpConnName`
		`java.lang.String`
		The connection name in the SP connection that bridges the service provider.

IdP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubSpConnPartnerId`
		`java.lang.String`
		The **Partner's Entity ID** in the SP connection that bridges the service provider.

IdP connection	SAML 2.0
	SAML 1.x
	WS-Federation
		`#FedHubSpConnProtocol`
		`java.lang.String`
		The protocol of the IdP connection. The returned values are `SAML20`, `SAML11`, `SAML10`, or `WSFED`.

Not applicable	OAuth	`#FedHubOAuthClientId`
		`java.lang.String`
		The client ID in the authorization server that bridges the service provider.

Not applicable	OAuth	`#FedHubOAuthClientName`
		`java.lang.String`
		The client name in the authorization server that bridges the service provider.

Sample customizations

Use OGNL expressions to customize assertions and authentication requests in different ways.

Add SessionNotOnOrAfter to assertions

This expression adds the optional `SessionNotOnOrAfter` attribute to the `<AuthnStatement>` element and sets the value to 60 minutes.

Message Type

`AssertionType`

Expression

```
#cal = new org.apache.xmlbeans.XmlCalendar(new java.util.Date()),
#cal.setTimeZone(@java.util.TimeZone@getTimeZone("UTC")),
#cal.add(@java.util.Calendar@MINUTE, 60),
#AssertionType.getAuthnStatementArray(0).setSessionNotOnOrAfter(cal)
```

Expected assertions

```
...
<saml:AuthnStatement ... AuthnInstant="2015-03-20T16:27:37.344Z"
 SessionNotOnOrAfter="2015-03-20T17:27:37.398Z">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>...</saml:AuthnContextClassRef>
    </saml:AuthnContext>
</saml:AuthnStatement>
...
```

Use well-formed XML as attribute value

The following expression inserts well-formed XML in the `<AttributeValue>` element if the **Attribute Name Format** is `urn:pingidentity.com:SAML:attrname-format:xml:complex`.

Message Type

`AssertionType`

Expression

```
#i = 0,
#AssertionType.getAttributeStatementArray(0).getAttributeArray().
{#this.getNameFormat().equals('urn:pingidentity.com:SAML:attrname-format:xml:complex')?
{#AssertionType.getAttributeStatementArray(0).getAttributeArray(i).removeAttributeValue(1)}:null,
#i = #i+1}
```

> **Note**
>
> Line breaks are inserted for readability only. Statements calling methods whose arguments are enclosed in quotes must be entered on a single line.

This example uses well-formed XML as the attribute value for attributes that are configured as `urn:pingidentity.com:SAML:attrname-format:xml:complex` (a custom attribute name format added to `<pf_install>/pingfederate/server/default/data/config-store/custom-name-formats.xml`) in the **Attribute Contract** window. You can use other application logic here.

Sample inputs (attributes and their values)

Example input 1

Attribute Name	ExtAttr1
Attribute Name Format	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified
Attribute Value	`123`

Example input 2

<table>
<tr><td>Attribute Name</td><td>ExtAttr2</td></tr>
<tr><td>Attribute Name Format</td><td>urn:pingidentity.com:SAML:attrname-format:xml:complex</td></tr>
<tr><td>Attribute Value</td><td><pre><code><saml:Attribute
 xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
 Name="ExtAttr2"
 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml:AttributeValue
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:customNs="http://www.sample.tld/customnamespace">
        <customNs:Line>Documentation</customNs:Line>
        <customNs:Line>Ping Identity</customNs:Line>
    </saml:AttributeValue>
</saml:Attribute></code></pre>
ⓘ <b>Note</b>
This is a well-formed XML document in one line.</td></tr>
</table>

Expected results

```
...
<saml:Attribute Name="ExtAttr1"
 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml:AttributeValue xsi:type="xs:string"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        123
    </saml:AttributeValue>
</saml:Attribute>
<saml:Attribute Name="ExtAttr2"
 NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
    <saml:AttributeValue
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:customNs="http://www.sample.tld/customnamespace">
        <customNs:Line>Documentation</customNs:Line>
        <customNs:Line>Ping Identity</customNs:Line>
    </saml:AttributeValue>
</saml:Attribute>
...
```

Include extensions in authentication requests

This expression includes the optional `Extensions` element in the authentication requests if a certain query parameter (`oid` in this example) is sent to the `/sp/startSSO.ping` endpoint to start an SP-initiated SSO request.

Message Type

`AuthnRequestDocument`

Expression

```
#element = #XmlHelper.addToSaml2Extensions(#AuthnRequestDocument, '<samplens:orgId name="orgId"
xmlns:samplens="urn:org.sample.wms"/>'),
#value = #HttpServletRequest.getParameter('oid') == null ? 'someDefaultValue' :
#HttpServletRequest.getParameter('oid') ,
#XmlHelper.setAttribute(#element, 'value', #value)
```

Expected AuthnRequest

A GET request to https://<*pf_host*>:<*pf.https.port*>/sp/startSSO.ping?PartnerIdpId=<*entityID*>&oid=123 would trigger the following Extensions block.

```
<samlp:AuthnRequest ...>
  <saml:Issuer ...>...</saml:Issuer>
  <samlp:Extensions>
    <samplens:orgId name="orgId" value="123" xmlns:samplens="urn:org.sample.wms"/>
  </samlp:Extensions>
  ...
</samlp:AuthnRequest>
```

For information about OGNL, see the Apache Commons OGNL Language Guide.

Fulfillment by datastore queries

You can configure PingFederate to use local datastores to fulfill various contracts or conditions to verify in the token authorization process.

Datastores represent external systems where user attributes and other data are stored. Once defined, you can configure PingFederate to retrieve user attributes from datastores for contract fulfillment and token authorization in various use cases.

PingFederate supports a wide variety of database servers and directory servers. As needed, the PingFederate SDK supports the creation of custom drivers for connecting to other types of data repositories, such as flat files or SOAP-connected databases. For more information, see the Javadoc for the `CustomDataSourceDriver` interface, the `SamplePropertiesDataStore.java` file for a sample implementation, and the SDK Developer's Guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

Related links

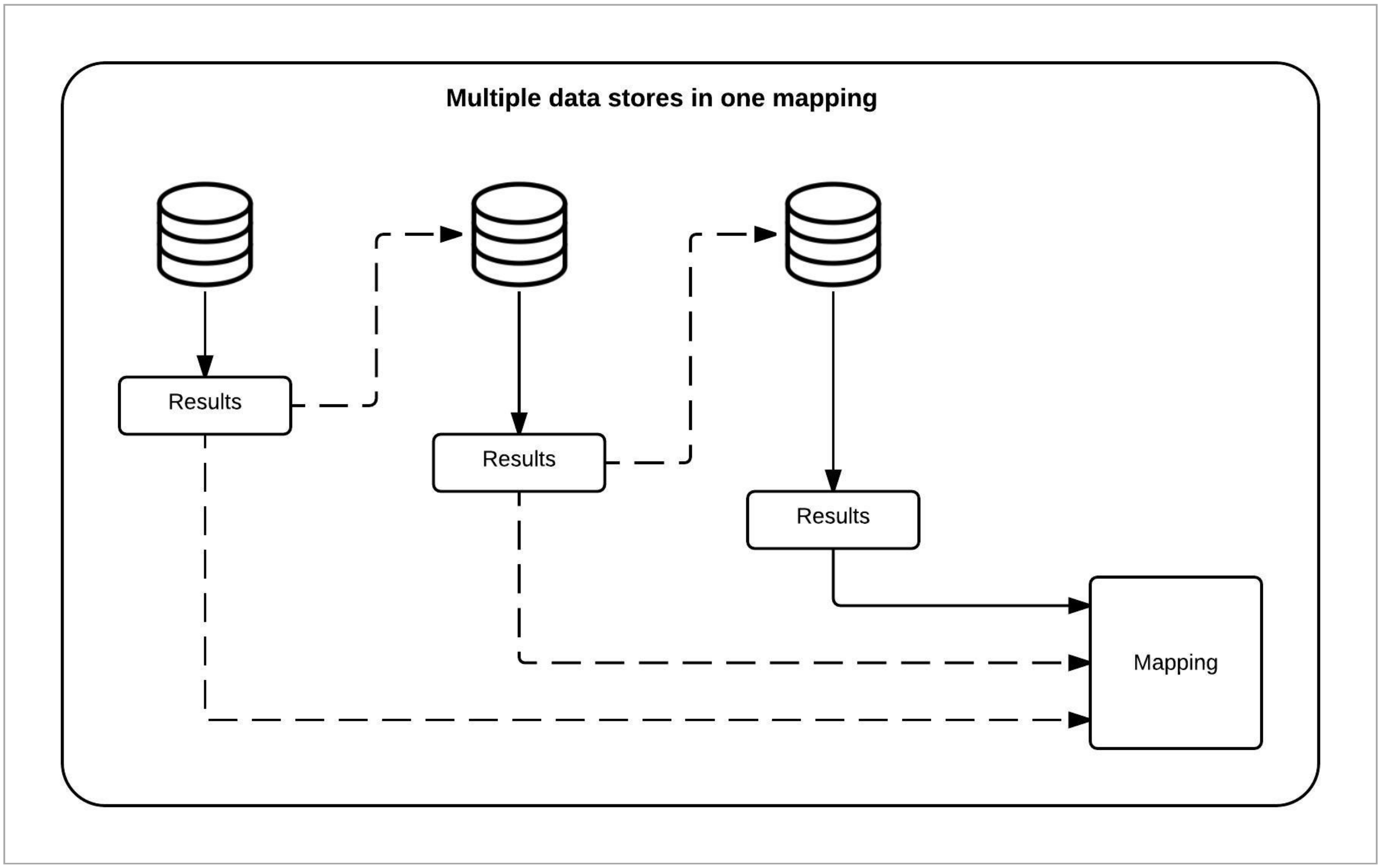
- Datastores
- System requirements

Attribute mapping with multiple data sources

PingFederate can query multiple datastores for additional attributes in most configurations.

Multiple datastores in one mapping

The PingFederate IdP server supports separate datastores to look up attributes for a single mapping. For example, you can query multiple datastores for values and map those values in one mapping, or query a datastore for a value and use that value as input for subsequent queries of other datastores.



If a datastore uses results from previous queries as input, and if the previous queries return no result, PingFederate continues the process by moving on to the next datastore in the setup. If you prefer PingFederate to abort and fail the requests, see Configuring the behavior of searching multiple datastores with one mapping.

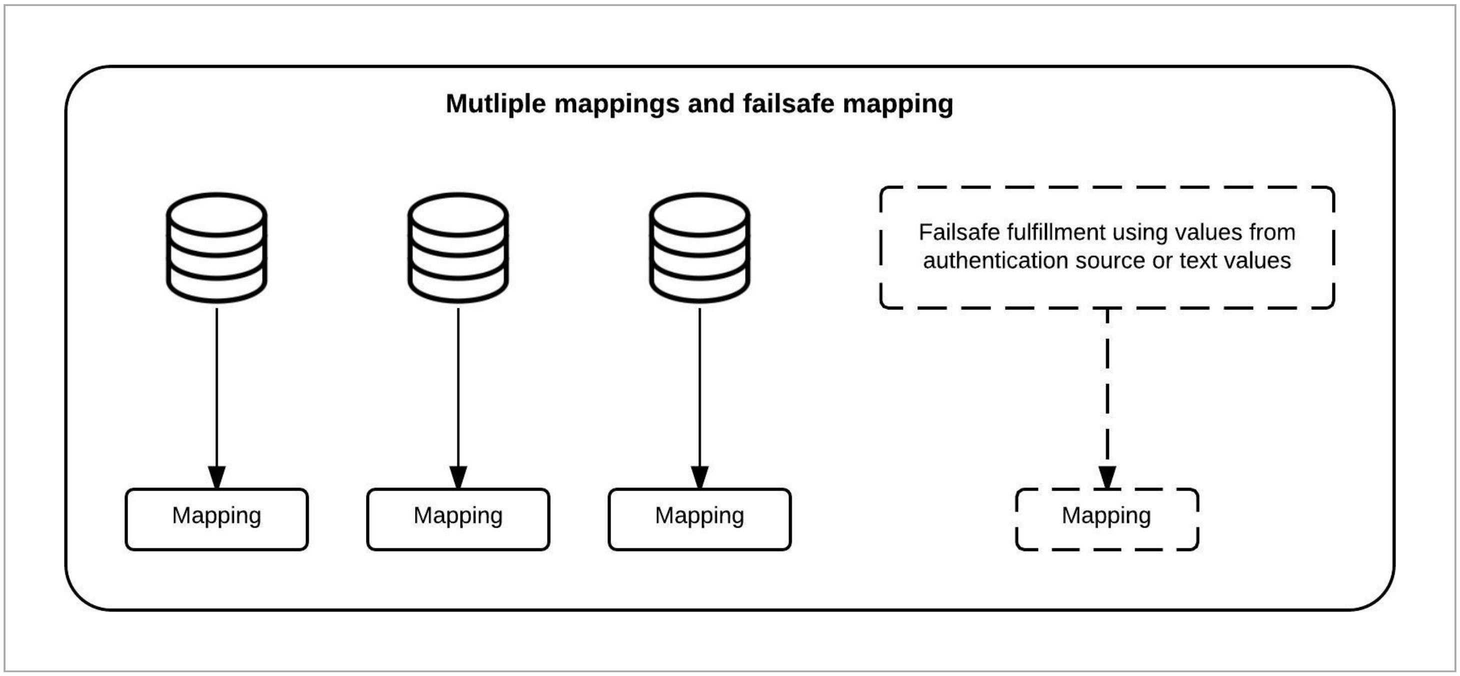
If a required attribute, such as `SAML_SUBJECT` in a SAML contact or `subject` in an authentication policy contract, is not fulfilled, the request fails.

Multiple datastores in one mapping are available for browser single sign-on (SSO) and WS-Trust security token service (STS) on the identity provider (IdP) side, authentication policy contract to service provider (SP) adapter mappings, and the following OAuth workflows:

- Identity provider (IdP) adapter grant mappings
- Resource owner credential grant mappings
- Access token mappings
- OpenID Connect policies (the user-attributes mapping process)

Multiple mappings and failsafe mapping

For browser SSO and WS-Trust STS on the IdP side, PingFederate also supports separate search parameters for each datastore and for "fall-through" searches. For example, you can add the same datastore more than once, using different search queries for each instance, or you can search different datastores successively. If any search fails to find a user in the specified attribute source, the next search is executed until a match is found. You can also configure a failsafe attribute source, providing a default set of attributes from the IdP adapter and using text values.


Mutliple mappings and failsafe mapping
Failsafe fulfillment using values from authentication source or text values
Mapping
Mapping
Mapping
Mapping

Datastore query configuration

You can query attributes from a datastore by choosing a datastore instance, specifying the search location and the desired attributes, and entering the search term.

After the configuration is complete, you can fulfill a contract or verify a condition in the Token Authorization framework using the results from the datastore queries.

Choosing a datastore

This topic describes how to select a datastore for PingFederate to use for looking up attributes.

About this task

On the **Data Store** window, choose a datastore for PingFederate to look up attributes.

Steps

1. Enter a description and ID, if prompted, for the datastore.
2. Select a datastore instance from the **Active Data Store** list.

   > **Tip**
   >
   > If the datastore you want is not shown in the **Active Data Store** list, click **Manage Data Stores** to review or add a datastore instance.

3. Depending on the datastore type, the rest of the setup varies as follows.

Data store type	Required tasks
JDBC	◦ Specifying database tables and columns ◦ Entering a database search filter
LDAP	◦ Specifying directory properties and attributes ◦ Defining encoding for binary attributes (optional) ◦ Entering a directory search filter
Other	◦ Specifying data source filters and fields

Specifying database tables and columns

In the **Database Table and Columns** window, you can specify exactly where additional data can be found to fulfill your use case. You can only use one table as a source of data for a database query.

About this task

For more information about each field, see the following table.

Field	Description
Schema	Lists the table structure that stores information within a database. Some databases require selection of a specific schema for database queries. Other databases do not require selection of a schema.
Table	Displays the tables contained in the database. Select the table to retrieve data from the datastore.
Columns to return from SELECT	Displays selected columns from the selected tables. Select the columns that are associated with the desired attributes you want to return from the database queries.

Important

For MySQL users — To allow for table and column names that might contain spaces, PingFederate inserts double quotes around the names at runtime. To avoid SQL syntax errors resulting from the quotes, add the session variable `sql_mode=ANSI_QUOTES` to the Java Database Connectivity (JDBC) connection string of your datastore instance, as in the following example.

```
jdbc:mysql://myhost.mydomain.com:3306/pf?sessionVariables=sql_mode=ANSI_QUOTES
```

Alternatively, you can configure the system variable *sql_mode* with the `ANSI_QUOTES` option. For more information, see https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html.

Steps

1. From the **Schema** list, choose a schema when applicable.

2. From the **Table** list, select a table.

3. **Optional:** To determine what attributes to examine, click **View Attribute Contract**.

4. **Optional:** To update an existing configuration where changes to the database might have occurred, click **Refresh**.
5. Under **Columns to return from SELECT**, choose a column name and click **Add Attribute**.

   > **Note**
   >
   > You do not have to add a column here to use it as part of a search filter. Add only the column that is required by subsequent sibling configuration items, such as contract fulfillment or token authorization. Any added columns left unused are removed when the configuration is saved.

   Repeat this step to add more columns as needed.

Example

Suppose you, the identity provider (IdP), have a data table named **ACCESSTABLE** with thee columns: `userid`, `department`, and `accesslevel`. Your use case requires you to map `accesslevel` into a SAML contract to an SP.

On the **Database Table and Columns** window, select the **ACCESSTABLE** table and add the `accesslevel` column.

Entering a database search filter

Use a `WHERE` clause on the **Database Filter** window to set up a database search filter in PingFederate.

About this task

On the **Database Filter** window, enter a `WHERE` clause for PingFederate to query the database table you selected to retrieve a record associated with particular values. The clause is in the form:

`[WHERE] column1=value1`

The left side (*column1*) is a column from the database table that you selected on the **Database Table and Columns** window.

> **Tip**
>
> To get a list of columns, click the **View List of Columns from ...** link.

The right side (*value1*) is the match-against value, generally a variable passed in from either an authentication source for an identity provider (IdP) or an assertion for a service provider (SP). The variables are shown underneath the **Where** text field. If you are retrieving attributes from multiple data stores using one mapping, attributes available from other sources, if previously configured, are listed near the bottom of the window.

You can also apply additional search criteria by using other columns from the targeted table.

Steps

1. Enter a *WHERE* clause in the text field.

   > **Note**
   >
   > The initial `WHERE` is optional.

2. Ensure the syntax and variable names are correct. For more information about *WHERE* clauses, consult your database management system (DBMS) documentation.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

3. Click **Next** to complete the configuration to query attributes from the database server.

   Later in the workflow, you can use the attribute values returned from the database in the applicable contract fulfillment window, the issuance criteria window, or both, to fulfill your use case.

Example

Suppose you have selected a data table named **ACCESSTABLE** on the **Database Table and Columns** window. You, the IdP, want to locate user records by matching `userid` column against the username from an HTML Form Adapter. As a passed-in variable from the HTML Form Adapter, `${username}` is shown underneath the **Where** text field.

On the **Database Filter** window, enter the following filter in the **Where** text field:

```
userid='${username}'
```

`userid`

The column in the table containing the username information in this example.

`'${username}'`

The value of the username variable (`username`) from an HTML Form Adapter

> **Important**
>
> You must use the `$\{}` syntax to retrieve the value of the enclosed variable and insert single quotation marks around the `$\{}` characters.

Specifying directory properties and attributes

Use these instructions to initiate ways to specify methods for PingFederate to search for particular user data.

About this task

On the **LDAP Directory Search** window, specify the branch of your directory hierarchy where you want PingFederate to look up user data. For more information about each field, refer to the following table.

Field	Description
Base DN	The base distinguished name (DN) of the tree structure in which the search begins. This field is optional if records are located at the root of the directory.
Search Scope	The node depth of the query. Select **Subtree** (the default value), **One level** or **Object**.
Root Object Class	The object class containing the desired attributes.

Field	Description
Attributes	A list of attributes based on the selected **Root Object Class** value.
Option (optional)	The attribute option for the selected attribute.

Steps

1. **Optional:** Specify a base DN.

   > **Tip**
   >
   > Choose a base DN that is as specific as possible for your search. A broad base DN can result in longer search times and increased network traffic, while a narrow base DN can help ensure that your search is accurate and efficient.

2. Select a search scope.
3. **Optional:** Click **View Attribute Contract** to determine what attributes to look up.
4. Select a root object class, an attribute, and, optionally, enter an **Option**. Click **Add Attribute**.

   > **Note**
   >
   > You do not have to add an attribute here to use it as part of a search filter. Add only the attributes that are required by subsequent sibling configuration items, such as contract fulfillment or token authorization. Any added attributes that are left unused are removed when the configuration is saved.

   Choose from:

   - Microsoft Active Directory

     If you choose the `memberOf` attribute, an optional check box, **Nested Groups**, appears on the right. Select this check box if you want PingFederate to query for groups the end users belong to directly and indirectly through nested group membership (if any) under the base DN.

     For example, if you have three groups under a base DN: Canada, Washington and Seattle. Seattle is a member of Washington. Ana Smith is an end user and a member of Seattle. If the **Nested Groups** check box is selected, when PingFederate queries for Ana's `memberOf` attribute values, the expected results are Seattle and Washington. When the **Nested Groups** check box is not selected (the default), the expected result is Seattle.

     > **Important**
     >
     > Do not enter any value for the **Option** field. Only the attributes that are defined in the directory server schema can be returned.

   - Oracle Directory Server or Oracle Unified Directory

     Choose `isMemberOf` under **Attribute** for nested group membership. For information related to Oracle Directory Server, go to docs.oracle.com/cd/E29127_01/doc.111170/e28967/ismemberof-5dsat.htm. For information related to Oracle Unified Directory, go to Fusion Middleware Administering Oracle Unified Directory and search for *memberof user attributes*.

> **Tip**
>
> If you need to include `tokenGroups` as one of the attributes, select **Object** as the search scope and enter a base DN matching the subject DN of the authenticated user—you can use variables from the authentication source (an adapter or an authentication policy contract) or results from the previous lookup in the base DN to fulfill this requirement.

5. Repeat step 4 to add more attributes as needed.

Example

Suppose you want to map the `sn` Active Directory (AD) user attribute into an OpenID Connect policy. The users for this use case reside under a specific container on your directory server, `OU=West, DC=example, DC=com`.

On the **LDAP Directory Search** window, enter `OU=West, DC=example, DC=com` as the base DN, keep the default **Search Scope** value (**Subtree**), select **<Show All Attributes>** from the **Root Object Class** list, select the `sn` AD user attribute, and click **Add Attribute**.

Defining encoding for binary attributes

Use the **LDAP Binary Attribute Encoding Types** window to specify an encoding type to apply during fulfillment.

About this task

The **LDAP Binary Attribute Encoding Types** window appears when at least one attribute is configured as such in the datastore. Because you cannot use binary attribute data in an assertion to the service provider (SP), you must specify the encoding type that you want to apply during fulfillment. The available choices are **Base64**, **Hex**, and **SID**.

> **Note**
>
> Defining encoding for binary attributes is only applicable to identity provider (IdP) and IdP-to-SP bridging use cases.

Steps

1. To set an encoding type, select a value from the **Attribute Encoding Type** list.

   Repeat this step for each binary attribute.

Example

Microsoft Office 365 relies on an immutable Active Directory binary attribute associated with user accounts (`objectGUID`), and requires this binary data to be Base64-encoded to correlate provisioned federated user data to Active Directory accounts. Select **Base64** from the **Attribute Encoding Type** list.

Claims-based authentication with Microsoft Outlook Web App and Exchange admin center (EAC) requires `tokenGroups` (another binary attribute in Active Directory) to be SID-encoded. Select **SID** from the **Attribute Encoding Type** list.

Entering a directory search filter

You can use a filter in PingFederate to query your selected data and retrieve a record associated with it.

About this task

On the **LDAP Filter** window, enter a filter for PingFederate to query the data you selected. The filter is in the form:

```
attribute1=value1
```

The left side (*attribute1*) is an attribute from your directory.

> **Tip**
>
> To see a list of attributes, click the **View List of Available LDAP Attributes** link.

The right side (*value1*) is the match-against value, generally a variable passed in from either an authentication source for an identity provider (IdP) or an assertion for a service provider (SP). The variables are shown underneath the **Filter** text field. If you are retrieving attributes from multiple data stores using one mapping, attributes available from other sources, if previously configured, are listed near the bottom of the window.

You can also apply additional search criteria by using other attributes from the target object class.

A filter narrows a search to locate requested data by either including or excluding specific records. A filter includes the attributes in the search and the value or range of values that the search is attempting to match. Searches are conducted by using three components: at least one attribute (attribute data type) on which to search, a search filter operator that will determine what to match, and the value of the attribute being sought.

Steps

1. On the **LDAP Filter** window, enter a search filter in the text field.

   > **Tip**
   >
   > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Ensure the syntax and variable names are correct. For general information about search filters, consult your directory documentation.

3. Click **Next** to complete the configuration to query attributes from the directory server.

   Later in the workflow, you can use the attribute values returned from your directory server in the applicable contract fulfillment window, the issuance criteria window, or both, to fulfill your use case.

Example

Suppose you want to locate user records by matching the `mail` Active Directory (AD) user attribute against an extended attribute, `eml`, in your access token contract for the purpose of mapping attributes to an OpenID Connect policy. As a passed-in variable from the access token, `${eml}` is shown underneath the **Filter** text field.

On the **LDAP Filter** window, enter the following filter in the **Filter** text field.

```
mail=${eml}
```

`mail`

An AD user attribute containing the email address of the user

`${eml}`

The value of the extended attribute (`eml`) in the access token contract

> **Important**
>
> You must use the `$\{}` syntax to retrieve the value of the enclosed variable.

Specifying data source filters and fields

PingFederate lets you configure data source filters, fields, and other settings for your datastore.

When configuring how PingFederate will retrieve attribute values from datastores, the procedures depend on the type of datastore.

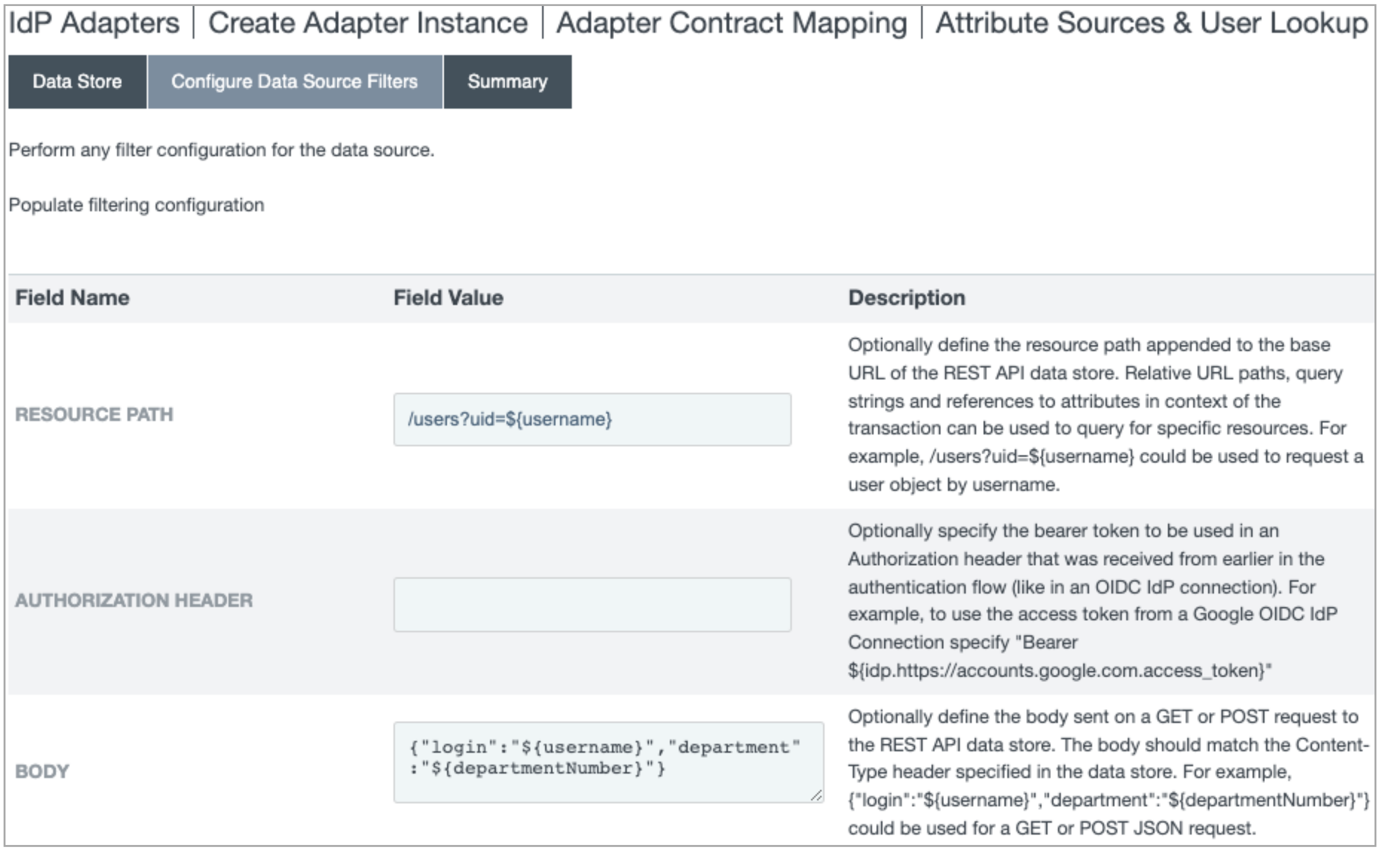
- Specifying data source filters for REST API datastores
- Specifying a dynamic authorization header for a REST API datastore
- Specifying filters and fields for a custom datastore
- Specifying filters and fields for an AWS DynamoDB datastore

Specifying data source filters for REST API datastores

On the **Configure Data Source Filters** window, you can specify a resource path for a REST API datastore.

Steps

1. On the **Attribute Sources & User Lookup** window, go to the **Configure Data Source Filters** tab.



IdP Adapters | Create Adapter Instance | Adapter Contract Mapping | Attribute Sources & User Lookup
Data Store
Configure Data Source Filters
Summary

Perform any filter configuration for the data source.

Populate filtering configuration

Field Name	Field Value	Description
RESOURCE PATH	/users?uid=${username}	Optionally define the resource path appended to the base URL of the REST API data store. Relative URL paths, query strings and references to attributes in context of the transaction can be used to query for specific resources. For example, /users?uid=${username} could be used to request a user object by username.
AUTHORIZATION HEADER		Optionally specify the bearer token to be used in an Authorization header that was received from earlier in the authentication flow (like in an OIDC IdP connection). For example, to use the access token from a Google OIDC IdP Connection specify "Bearer ${idp.https://accounts.google.com.access_token}"
BODY	{"login":"${username}","department":"${departmentNumber}"}	Optionally define the body sent on a GET or POST request to the REST API data store. The body should match the Content-Type header specified in the data store. For example, {"login":"${username}","department":"${departmentNumber}"} could be used for a GET or POST JSON request.

2. **Optional:** Enter a relative path, additional query parameters, or both in the **Resource Path** field.

> **Tip**
>
> You can specify a default attribute value in the form of `$\{attributeName:-defaultValue}` to use at runtime when the attribute value is not available. Do not use `$\{` and `}` in the default value.

3. **Optional:** To use a previously-received bearer token in an authorization header, specify the bearer token in the **Authorization Header** field.

4. **Optional:** Define the **Body** sent with a GET or POST request to the REST API datastore.

Example

You have use cases that can leverage user attributes obtained through REST APIs. The data source returns user records in JSON. It also provides the following paths to access its data based on user populations:

- https://rest.example.com/development/users
- https://rest.example.com/staging/users

To retrieve the record of a particular user, the request must include the `uid` query parameter with the identifier of the user.

Your use cases focus on users under the `/staging/users` path. Your authentication policy uses the HTML Form Adapter, which captures user identifiers using the `username` attribute.

To address this sample use case:

1. Create a REST API datastore with a base URL of https://rest.example.com.

2. Add the REST API datastore as an attribute source in the applicable use cases, such as a service provider (SP) connection that uses an HTML Form Adapter instance or an OAuth identity provider (IdP) Adapter Mapping configuration that maps from an HTML Form Adapter instance into the persistent grants.
3. When prompted to configure the filtering option on the **Configure Data Source Filters** window, enter `/staging/users?uid=${username}` in the **Resource Path** field.

Related links

- Configuring a REST API datastore

Specifying a dynamic authorization header for a REST API datastore

When you configure an authentication source with an application, you can use the access token from the connection as a bearer token in an authorization header to receive additional information as needed.

About this task

The access token used in the authorization header can originate from various upstream sources within the PingFederate policy, such as gIdP adapters that generate and retrieve access tokens or connections to external IdPs that provide access tokens upon successful authentication.

Before you begin

- Create a service provider (SP) or IdP connection.
- Configure an IdP authentication policy for the connection.

Steps

1. Get the access token that you plan to use as a bearer token.

   After you've made the connection, you can find the access token attribute name in `<pf_install>/pingfederate/log/server.log` in debug mode.

   > **Note**
   >
   > The access token name is generated based on your specific PingFederate configuration.

2. On the **Configure Data Source Filters** page, in the **Authorization Header** field, enter the access token attribute name.

   > **Tip**
   >
   > You can reference attribute values in the form of `${attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value isn't available. Do not use `${` and `}` in the default value.

Example

Authorization headers

Sample **Authorization Header** entries are shown here:

- For a Yahoo OpenID Connect Connection: `Bearer $\{idp.https://api.login.yahoo.com.access_token}`
- For a Google OpenID Connect Connection: `Bearer $\{idp.https://accounts.google.com.access_token}`
- For an adapter: `Bearer ${adapter.myadapterid.token}`

Specifying filters and fields for a custom datastore

PingFederate enables you to specify a filter/search query for your data source, then configure fields for your use case.

About this task

Follow this path to set up attribute queries from a custom datastore.

Steps

1. On the **Configure Data Source Filters** window, specify a filter, or search query, for your data source.

   This window display and the syntax of the filter depend on your developer's implementation of the PingFederate SDK.

2. On the **Configure Data Source Fields** window, select the fields applicable to your use case.

   These choices are supplied by the driver implementation. Select only those needed to fulfill your use case.

Result

You can use the values returned from your data source in the applicable contract fulfillment window, the issuance criteria window, or both.

Specifying filters and fields for an AWS DynamoDB datastore

PingFederate enables you to specify a filter or search query for your Amazon Web Services (AWS) DynamoDB datasource, then configure fields for your use case.

About this task

Specifying filters and fields in the DynamoDB datastore optimizes data retrieval, improves query performance, and enhances application responsiveness by narrowing down search results and selecting specific attributes.

To set up queries on an AWS DynamoDB datastore to use during attribute lookup:

Steps

1. In the **Configure Data Source Filters** window, specify a filter or search query for your data source.

   For information about each field, see the following table.

Field	Description
Index Name	The name of the DynamoDB index to perform the query on. Leave this field blank to query the base table.
Key Condition Expression	The key condition expression to use for the DynamoDB query. For more information, see Key condition expressions for query in the AWS DynamoDB documentation.
Filter Expression	Optional filter expression to define for the DynamoDB query. For more information, see Filter expressions for query in the AWS DynamoDB documentation.
Expression Attribute Values	The expression attribute values that map to the reference in the key condition expression and the filter expression. You can use references to attributes in the context of the transaction. For more information, see Expression attribute values in the AWS DynamoDB documentation.
Expression Attribute Names	Optionally define the expression attribute names to reference in the key condition expression and the filter expression. For more information, see Expression attribute names in the AWS DynamoDB documentation.

> **Note**
>
> Values with a `NULL` data type evaluate to a Boolean `true` result.

2. Click **Test Attribute Lookup** to test your query configuration.

   Result:

   - If you have correctly configured the query, <*Local Attribute Names*> and <*Test Result Values*> are returned.
   - If there is an error in the query, an error message is returned.

Configuring failsafe options

From the **Failsafe Attribute Source** window, you can enable the failsafe mapping or stop the single sign-on (SSO) transaction when all attribute sources fail to return values for any reason.

About this task

When a datastore is configured and the attribute mappings under **Attribute Sources & User Lookup** fail to complete the attribute contract in a service provider (SP) connection, you can choose to configure a set of failsafe **Attribute Contract Fulfillment** mappings. For example, you might configure a set of attributes to identify the SSO subject as a guest user at the SP.

Note

The **Failsafe Attribute Source** window does not appear if you have selected the **Retrieve additional attributes from multiple data stores using one mapping** option on the **Mapping Method** window.

Important

The attribute contract is fulfilled using either the mapping configured under **Attribute Sources & User Lookup** or the failsafe mapping, not both. In other words, you cannot use the failsafe mapping to fill in missing attributes when some are found with the datastore mapping setup but others are not.

The failsafe mapping is used only when all of the mappings configured in the datastore setup fail to return values for any reason. If any mapping succeeds (an attribute mapped to text, for example), failover does not occur.

Alternatively, you can have PingFederate stop the SSO transaction. This choice depends on your agreement with the SP.

Steps

- To enable the failsafe mapping, select **Send user to SP using default list of attributes**, and then map or enter the desired values on the **Attribute Contract Fulfillment** window.
- To stop the SSO transaction, select **Abort the SSO transaction**.

Reviewing datastore query configurations

You can review the configuration on the **Summary** window to determine whether to change your configuration, save those changes, or discard them.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

  Tip

  When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.
- To discard your changes, click **Cancel**.

IdP-to-SP bridging

The IdP-to-SP bridging links on the Server Configuration menu provide access to advanced federation settings.

Adapter-to-adapter mappings

PingFederate can act as both identity provider (IdP) and service provider (SP) running on the same server with this configuration.

This configuration is provided for special use cases in which PingFederate is acting as both an IdP and an SP, and user attributes from an IdP adapter are used to create an authenticated session with an SP adapter on the same PingFederate server. Generally, these cases involve software-as-a-service (SaaS) providers who might not support standards-based single sign-on (SSO) but do provide proprietary SSO with "delegated authentication", such as Salesforce and Workday.

In effect, this configuration provides an alternative to setting up complete connections to send SAML assertions and other messages back and forth between an IdP and an SP running on the same PingFederate server in a loop-back configuration to enable nonstandard use cases. Instead, attributes that would normally be sent in an assertion are mapped directly from the IdP authentication adapter to an SP adapter, resulting in a secure SP user session.

To use this configuration, ensure that you have already configured the required IdP and SP adapter instances. You can reuse instances that are also in use for connection configurations.

Managing mappings

Manage adapter-to-adapter mappings using the **Adapter-to-Adapter Mappings** window to control how many you use in PingFederate.

About this task

On the **Adapter-to-Adapter Mappings** window you can add, modify, or delete adapter-to-adapter mappings. The **Adapter-to-Adapter Mappings** window is located at **Applications > Integration > Adapter-to-Adapter Mappings**.

Steps

- To add a mapping, select an identity provider (IdP) adapter instance from the **Source Instance** list, a service provider (SP) adapter instance from the **Target Instance** list, and then click **Add Mapping**.

  > **Note**
  >
  > You can create only one mapping of a source to the same target. However, you can map different sources to the same target, and the reverse.

- To edit a mapping, select the mapping and then follow the configuration wizard to complete the task.
- To remove a mapping, select **Delete** under **Action** for the mapping.

Assigning a license group

Select a license group if your PingFederate license manages connections by groups.

About this task

Adapter-to-adapter mapping is considered a connection for licensing purposes.

> **Note**
>
> This window is not displayed for unrestricted or other types of licenses.

Steps

- Select the license group from the list.

Identifying the target application

On the **Target App Info** tab, you can enter the name of the target application and the URL of the application icon.

About this task

The URL is accessible through the IdP Adapter interface, `IdpAuthenticationAdapterV2`, in the PingFederate Java SDK. For more information about the SDK, see SDK Developer's Guide. Both fields are optional.

> **Note**
>
> If this is a child instance, select the **Override** checkbox to modify the configuration.

Steps

1. **Optional:** Enter the application name in the **Application Name** field.
2. **Optional:** Enter the URL to the application icon in the **Application Icon URL** field.
3. Click **Next**.

Configuring attribute sources and user lookup for adapter-to-adapter mappings

To look up attributes and to set up search parameters, configure or modify attribute sources to configure one or more datastores.

About this task

Attribute sources are specific datastore or directory locations containing information possibly required to fulfill the service provider (SP) adapter contract.

Steps

- To configure an attribute source, click **Add Attribute Source** and complete the setup steps (see Choosing a datastore).
- To modify an attribute source configuration, select the attribute source and follow the configuration wizard to complete the task.

  > **Note**
  >
  > Depending on what you change, you might need to modify dependent data in subsequent steps, as indicated.

Configuring target application information

Optionally, you can specify the name and the icon URL of the target application in the **Target App Info** tab.

Steps

1. Enter the target application's name and icon URL.
2. Click **Next**.

Configuring contract fulfillment for adapter-to-adapter mappings

Use one of the listed sources to map each attribute to fulfill the adapter contract.

About this task

The next step in this configuration is to map values from the identity provider (IdP) adapter into the attributes required by the service provider (SP) adapter (the adapter contract).

On the **Adapter Contract Fulfillment** tab, map each attribute to fulfill the adapter contract from one of these sources:

Adapter

When you make this selection, the associated **Value** drop-down list is populated by the IdP adapter.

Context

Values are returned from the context of the transaction at runtime.

> **Note**
>
> The HTTP Request selection is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.
> Choose **Expression** and then click **Edit** to enter an expression (see Using the OGNL edit window). If the **Expression** selection is not listed, then the feature is not enabled (see Enabling and disabling expressions). For syntax and examples, see sections under Construct OGNL expressions.

LDAP/JDBC/Other (when a datastore is used)

Values are returned from your datastore, if used. When you make this selection, the **Value** list is populated by the attributes from the datastore.

Expression (when enabled)

This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see Attribute mapping expressions). All of the variables available below for text entries are also available for expressions.

No mapping

Select this option to ignore the **Value** field, making the value selection unnecessary.

Text

The value is what you enter. This can be text only, or you can mix text with references to any of the values from the IdP adapter, using the `${attribute}` syntax.

The value is what you enter. This can be text only, or you can mix text with references to any of the values from the IdP adapter, using the `${attribute}` syntax.

You can also enter values from your datastore, when applicable, using the following syntax.

```
${ds.attr-source-id.attribute}
```

where `attr-source-id` is the **Attribute Source ID** value (see Choosing a datastore) and attribute is any of the datastore attributes you select.

Steps

1. Select a source of the attribute under **Source** for each SP adapter contract attribute.
2. Specify an attribute under **Value** for each attribute.

Configuring a default target URL (optional)

You can enter a URL in the **Default Target URL** field to override any **SP Default URL SSO** setting.

About this task

For more information, see Configuring default URLs.

> **Note**
>
> If specified, the adapter-to-adapter endpoint parameter tab, enter your default URL into the the `TargetResource` overrides any default target URL for an adapter-to-adapter mapping. See If specified, the adapter-to-adapter endpoint parameter System-services endpoints.

Steps

1. On the **Default Target URL**If specified, the adapter-to-adapter endpoint parameter tab, enter your default URL into the the **Default Target URL** tab.

   > **Note**
   >
   > This URL overrides any setting within the **SP Default URL SSO** field.

2. Click **Next**.

Defining issuance criteria for adapter-to-adapter mappings

You can define the criteria that must be satisfied for PingFederate to further process a request.

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains …** or **multi-value does not contain …** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. On the **Issuance Criteria** tab, from the **Source** list, select the attribute’s source.

2. In the **Source** list, select the attribute’s source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

+

Source	Description
Adapter	Select to evaluate attributes from the IdP adapter instance.
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

+

1. From the Attribute Name list, select the attribute to be evaluated.

2. In the Condition list, select the comparison method.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**

- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

3. In the **Value** field, enter the comparison value.

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

4. In the **Error Result** field, enter a custom error message

Error results are handled in one of the two ways.

Redirect

When an `InErrorResource` URL is provided, the value of the **Error Result** field is used by the query parameter `ErrorDetail` in the redirect URL.

Template

When an `InErrorResource` URL is not provided, the value of the **Error Result** field is used by the variable *$errorDetail* in the `idp.sso.error.page.template.html` template file.

Using an error code in the **Error Result** field allows the error template or an application to process the code in a variety of ways, such as displaying an error message or e-mailing an administrator.

To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

5. Click **Add**.

6. **Optional**: Repeat to add more criteria.
7. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

- Token authorization
- Attribute mapping expressions
- Policies

Reviewing the adapter-to-adapter mapping

Use the **Summary** tab to edit, save, and discard changes to your configuration of an adapter-to-adapter mapping.

About this task

When you have finished configuring an adapter-to-adapter mapping, you can review the configuration on the **Summary** tab.

Steps

1.
   - To keep your changes, click **Save**.
   - To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
   - To discard your changes, click **Cancel**.

Token translator mappings

Use the token translator configuration to map incoming user attributes from an identity provider (IdP) token processor directly to a service provider (SP) token generator.

This configuration is provided for use cases in which the PingFederate WS-Trust STS exchanges one type of security token for another without needing to generate a SAML token in between. See WS-Trust STS. Use this configuration, for example, to convert a user's Kerberos token to a third-party proprietary web access management (WAM) session token.

In effect, this configuration provides an alternative to setting up complete STS connections to make such an exchange using the same instance of PingFederate. Instead, incoming user attributes from an IdP token processor are mapped directly to an SP token generator.

To use this configuration, ensure that you have enabled both the IdP and SP roles for PingFederate, including the WS-Trust protocol. See Enabling the WS-Trust protocol. Make sure to configure the required token-translator instances. You might reuse instances that are also in use for STS connection configurations.

Managing token mappings

Use the **Token Generator Mappings** window to manage your token-to-token mappings.

About this task

On the **Applications** > **Token Exchange** > **Token Translator Mappings** window you can add, modify, or delete token-to-token mappings.

Steps

- To add a mapping, select a token processor instance from the **Source Instance** list and a token generator instance from the **Target Instance** list, and then click **Add Mapping**.

  > **Note**
  >
  > You can create only one mapping of a source to the same target. However, you can map different sources to the same target, and vice versa. When you configure multiple identity provider (IdP) token processors, service provider (SP) token generators, or both, you must provide the `TokenProcessorId` , the `TokenGeneratorId` query parameters, or both, in the request (see System-services endpoints).

- To edit a mapping, select the mapping and then follow the configuration wizard to complete the task.
- To remove a mapping, select **Delete** under **Action** for the mapping.

Configuring attribute sources and user lookup for token mapping

Configure one or more datastores using this optional adapter-to-adapter configuration to look up attributes and to set up search parameters to find information needed to fulfill contracts.

About this task

Attribute sources are specific datastore or directory locations containing information that you might require to fulfill the contract of the token generator.

Steps

- To configure an attribute source, click **Add Attribute Source** and complete the setup steps. For more information, see Choosing a datastore.

- To modify an attribute source configuration, select the attribute source and follow the configuration wizard to complete the task.

> **Note**
>
> Depending on what you change, you might need to modify dependent data in subsequent steps, as indicated.

Configuring contract fulfillment for token exchange mapping

Use the listed sources to map values from the token processor into the attributes required by the token generator, the Token Generator Contract.

About this task

Map each attribute to fulfill the Token Generator Contract from one of these sources:

Token

When you make this selection, the associated **Value** drop-down list is populated by the token processor.

LDAP/JDBC/Other (when a datastore is used)

Values are returned from your datastore, if used. When you make this selection, the **Value** list is populated by the attributes from the datastore.

Expression (when enabled)

This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see Attribute mapping expressions). All of the following variables available for text entries are also available for expressions.

No Mapping

Select this option to ignore the **Value** field, making it necessary to have no value selection.

Text

The value is what you enter. This can be text only, or you can mix text with references to any of the values from the token processor, using the `$\{attribute}` syntax.

You can also enter values from your datastore, when applicable, using the following syntax.

```
$\{ds.attr-source-id.attribute}
```

where `attr-source-id` is the **Attribute Source ID** value (see Fulfillment by datastore queries) and `attribute` is any of the datastore attributes you select.

Steps

1. Under **Source**, select a source of the attribute for each attribute in the contract of the token generator.

2. Under **Value**, specify an attribute for each attribute.

Defining issuance criteria for token translator mapping

Use the PingFederate issuance criteria features to process requests based on individual attributes.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains …** or **multi-value does not contain …** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

+

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The **HTTP Request** and **STS SSL Client Certificate Chain** context values are retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

Source	Description
Token	Select to evaluate attributes from the token processor instance.

1. Select the attribute to evaluate under **Attribute Name**.

> **Note**
>
> To evaluate the **STS Basic Authentication Username**, **STS SSL Client Certificate Chain**, or **STS SSL Client Certificate’s Subject DN** context value, ensure that the associated authentication is enabled and configured on **System > Server > Protocol Settings** to open the **WS-Trust STS Settings** window.

Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value …** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+

> **Note**
>
> To evaluate the **STS SSL Client Certificate’s Subject DN** context value, you must select one of the **… DN** conditions. These methods normalize the DN before comparison to accommodate for different string representations that are still considered equivalent, such as case sensitivity or white space.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ The **Error Result** field is used by the `faultstring` element for SOAP 1.1 and the `Reason/Text` element for SOAP 1.2. For more information on SOAP, see the World Wide Web Consortium's .w3.org/TR/2000/NOTE-SOAP-20000508///[^].

+ Using an error code in the **Error Result** field allows an application to process the code in a variety of ways, such as display an error message or e-mail an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

+ . **Optional:** Repeat to add more criteria.

+ . If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

+ For more information, see Attribute mapping expressions.

1. Click **Show Advanced Criteria**.
2. In the **Expression** field, enter the required expressions.
3. In the **Error Result** field, enter an error code or message.

   > **Note**
   >
   > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click **Add**.
5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing the token exchange mapping

Use the **Summary** tab to modify a configuration and then either save, edit, or discard the changes.

About this task

When you finish configuring a token exchange mapping, you can review the configuration on the **Summary** tab.

Steps

- To keep your changes, click **Save**.

- To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
- To discard your changes, click **Cancel**.

Identity provider SSO configuration

Identity providers (IdP) can use the PingFederate administrative console to configure local application-integration information and to manage connections to service provider (SP)-partner sites.

You only need one connection per partner, even if you are targeting more than one web application at the destination SP site.

While your entity ID is defined on the **Federation Info** tab of the **Protocol Settings** window, you can identify your organization differently using virtual server IDs on a per-connection basis. For more information, see Multiple virtual server IDs.

You can deploy an SP connection to bridge a service provider to one or more identity providers through authentication policy contracts. For more information, see Federation hub use cases.

> **Note**
>
> This topic applies to configuration settings needed for browser-based single sign-on (SSO). If you are using PingFederate exclusively as a security token service (STS), start with WS-Trust STS configuration.

IdP application integration settings

PingFederate uses identity provider (IdP) adapters to look up session information and authenticate users in integrated applications.

The integration of local applications with PingFederate is the essential "first-mile" configuration that allows end-users to access protected resources across domains. This process is facilitated through the use of application-integration kits and a robust SDK (see Bundled adapters and authenticators).

PingFederate uses IdP adapters such as the **HTML Form Adapter** to authenticate users at your site through applications or access-management systems. Go to **Authentication > Integration > IdP Adapters** to configure instances of IdP adapters.

For authentication sources and selectors that are capable of delegating end-user interactions to external web applications, you can create authentication applications to represent them.

You can also set a default URL to which users can be directed during single logout (SLO), and you can look up system endpoints that application developers at your site need to access PingFederate's single sign-on (SSO) and SLO services.

> **Note**
>
> If your PingFederate configuration enables the WS-Trust security token service (STS), the selections under **Integration** also include menu items for configuring plugin **Token Processors** and optionally **STS Request Parameters**, see IdP configuration for STS.

Managing IdP adapters

You must configure at least one instance of an identity provider (IdP) adapter to set up connections to service provider (SP) partners to manage and view your IdP adapter in the **IdP Adapters** window.

About this task

An IdP adapter looks up session information and provides user identification to PingFederate. PingFederate comes bundled with the PingID integration kit and the following adapters:

- Composite Adapter
- HTML Form IdP Adapter
- HTTP Basic IdP Adapter
- Identifier First Adapter
- Kerberos Adapter
- OpenToken IdP Adapter
- Passthrough IdP Adapter
- PingID Adapter
- PingOne MFA Adapter
- PingOne Protect Adapter
- PingOne Verify Adapter
- PingOne DaVinci Adapter
- Reference ID IdP Adapter
- X.509 Certificate IdP Adapter

Additional integration kits are available on the PingFederate download page on the **Add-ons** tab.

Steps

1. Go to **Authentication > Integration > IdP Adapters**.
2. In the **IdP Adapters** window, choose from the following options.

Option	Description
Configure a new instance	Click **Create New Instance**
Modify an existing instance	Click the name of instance in the **Instance Name** column
View the usage of an existing instance	Click **Check Usage** in the **Action** column on the instance's row

Option	Description
Remove an existing instance	Click **Delete** in the **Action** column on the instance's row

> **Note**
>
> By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
> If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System** > **Server** > **General Settings**.

Creating an IdP adapter instance

To create an adapter instance, you must choose an adapter type.

Steps

1. On the **Type** tab, configure the basics of this adapter instance:
   1. Enter the **Instance Name** and **Instance ID**.
   2. In the **Type** list, select the adapter type.
   3. Optional: In the **Parent Instance** list, select an existing type.

      If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

Configuring an IdP adapter instance

You can configure existing instances of identity provider (IdP) adapters.

About this task

Depending on the selected adapter, the **IdP Adapters** window presents different configuration parameters.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. Click the IdP adapter instance you want to configure.
3. Follow the in-product instructions to configure the adapter instance.

> **Note**
>
> If this is a child instance, select the **Override** checkbox to modify the configuration.

If you are configuring one of the bundled adapters, you can find configuration information in Bundled adapters or Integrate with PingID for PingFederate SSO.

If you are configuring an adapter from an integration kit, including any software as a service (SaaS) connector, go to Integration overview, locate the adapter's documentation, and configure the adapter instance accordingly.

Invoking IdP adapter actions

You can write identity provider (IdP) adapters to fulfill a variety of functions.

About this task

You can write adapters to:

- Provide configuration assistance
- Perform validation actions
- Generate parameters that might need to be set manually in a configuration file

Steps

- Follow the on-screen instructions to complete the actions required.

Extending an IdP adapter contract

You can use the **Extended Contract** tab to extend the contract of existing identity provider (IdP) adapters, especially those using the Composite Adapter.

About this task

If you are using the Composite Adapter, you must add attributes from the IdP adapter instances that comprise the composite configuration in this tab. For more information, see Composite Adapter.

If the adapter doesn't return values for the extended attributes, or if you prefer to fulfill them differently using datastore queries, dynamic text values, or results from OGNL expressions, you can define their fulfillment on the **Adapter Contract Mapping** tab. For more information, see Defining the IdP adapter contract.

> **Note**
>
> If this is a child instance, select the **Override** checkbox to modify the configuration.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**.
2. Click the name of the IdP adapter instance you want to modify.
3. On the **Create Adapter Instance** window, select the **Extended Contract** tab.
4. For each attribute that you want to add, enter the name of the attribute under **Extend the Contract** and click **Add**.
5. Click **Save**.

Setting pseudonym and masking options

You can set pseudonym and masking options to uniquely identify a user to your PingFederate SP partners.

Steps

1. Go to **Authentication > Integration > IdP Adapters** and click the identity provider (IdP) adapter instance that you want to change.

2. On the **Adapter Attributes** tab, do the following:

    1. (Optional) In the **Unique User Key Attribute** list, select an attribute to uniquely identify users signing on with this adapter.

        The attribute's value is used to identify user sessions across all adapters. **None** is selected by default.

        > **Note**
        >
        > If you choose a custom user key attribute, PingFederate uses the value of the attribute after the Adapter Contract Mapping (if any) has been evaluated. If you choose a custom user key attribute that is based on the username, configure the adapter's password credential validator (PCV) to trim spaces.

        > **Important**
        >
        > For the HTML Form Adapter, If you enabled the **Revoke Sessions after Password Change or Reset** option on the **IdP Adapter** tab, you cannot select **None** as the unique user key attribute. Doing so results in an error message.

    2. Select the checkbox under **Pseudonym** for the user identifier of the adapter and optionally for the other attributes, if available.

        This selection is used if any of your service provider (SP) partners use pseudonyms for account linking.

        > **Note**
        >
        > A selection is required whether or not you use pseudonyms for account linking. This allows account linking to be used later without having to delete and reconfigure the adapter. Ensure that you choose at least one attribute that is unique for each user, such as a user's email, to prevent assigning the same pseudonym to multiple users.

    3. Select the checkbox under **Mask Log Values** for any attributes whose values you want PingFederate to mask in its logs at runtime.

        > **Note**
        >
        > Masking is not applied to the unique user key attribute in the logs even though the attribute used for the key is marked as **Mask Log Values**.

    4. If you plan to use OGNL expressions to map derived values into outgoing assertions and want those values masked, select the **Mask all OGNL-expression generated log values** checkbox.

Defining the IdP adapter contract

You can modify the default identity provider (IdP) adapter contract settings using the **Adapter Contract Mapping** tab.

About this task

An IdP adapter contract is a contract that can be used to fulfill the attribute contract passed to your service provider (SP) partners. By default, PingFederate fulfills the IdP adapter contract with attribute values from the adapter.

You can optionally configure PingFederate to fulfill the IdP adapter contract with:

- Attribute values from local datastores
- Dynamic text values
- Results from OGNL expressions
- A combination of any of these

In addition, you can verify requests using the token authorization framework.

Steps

1. Go to **Authentication** > **Integration** > **IdP Adapters**
2. Click the **Instance Name** of the existing IdP adapter instance you want to configure.
3. Go to the **Adapter Contract Mapping** tab.

   > **Note**
   >
   > If this is a child instance, select the **Override Adapter Contract** check box to modify the configuration unless you have already selected the override option on the **Extended Contract** tab, in which case the **Override Adapter Contract** check box is automatically selected for you.

4. Click **Configure Adapter Contract**.
   - For information on **Attribute Sources & User Lookup**, see Defining attribute sources and user lookup.
   - For information on **Adapter Contract Fulfillment**, see Configuring IdP adapter contract fulfillment.
   - For information on **Issuance Criteria**, see Defining issuance criteria for IdP adapter contract.
5. On the **Summary** tab, click **Done** to save your adapter contract configurations, or click **Cancel** to discard them.

Defining attribute sources and user lookup

In the **Attribute Sources & User Lookup** window, you can add new attribute sources or manage existing attribute sources to supply attributes for the identity provider (IdP) adapter contract or the token authorization framework.

About this task

Attribute sources are specific datastore or directory locations containing information that might be needed for the IdP adapter contract or the token authorization framework. You can use more than one attribute source when mapping values to the IdP adapter contract.

For more information about token authorization, see Token authorization.

The PingFederate IdP server supports separate datastores to look up attributes for a single mapping. For example, you can query multiple datastores for values and map those values in one mapping, or query a datastore for a value and use that value as input for subsequent queries of other datastores.

Queries are executed in the order they are displayed on the **Attribute Sources & User Lookup** tab. Use the up and down arrows as needed to adjust the order.

> **Note**
>
> If a required attribute, such as the user identifier `username` for the HTML Form Adapter or `subject` for the OpenToken IdP Adapter, cannot be fulfilled, the request fails.

Steps

1. Go to **Authentication > Integration > IdP Adapters**

2. Click the name of the existing IdP adapter instance you want to configure in the **Instance Name** list.

3. Click the **Adapter Contract Mapping** tab.

   > **Note**
   >
   > If this is a child instance, select the **Override Adapter Contract** check box to modify the configuration unless you have already selected the override option in the **Extended Contract** tab, in which case the **Override Adapter Contract** check box is automatically selected for you.

4. Click **Configure Adapter Contract**.

   Choose from:

   - If your use case requires only dynamic texts or results from OGNL expressions without any attributes from local datastores, skip to Configuring IdP adapter contract fulfillment.

   - To add an attribute source, click **Add Attribute Source**.

   - To modify an existing instance, select it by name under **Description**.

   - To remove an existing instance or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

Configuring IdP adapter contract fulfillment

You can map values into the identity provider (IdP) adapter contract using the **Adapter Contract Fulfillment** tab.

Steps

1. Go to **Authentication > Integration > IdP Adapters**
2. Click the **Instance Name** of the existing IdP adapter instance you want to configure.
3. Go to the **Adapter Contract Mapping** tab.

> **Note**
>
> If this is a child instance, select the **Override Adapter Contract** check box to modify the configuration unless you have already selected the override option in the **Extended Contract** tab, in which case the **Override Adapter Contract** check box is automatically selected for you.

4. Click **Configure Adapter Contract**.
5. Select a source in the **Source** list and specify a **Value** to associate with it.

   For more information about the **Source** list and the possible **Values**, see the following table.

Source	Description
Adapter	Select **Adapter** to use the attribute value returned by the IdP adapter without customization.
Context	Select **Context** to return specific information from the request.
JDBC, LDAP, or other types of datastores (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a datastore. When you make this selection, the **Value** list is populated with attributes from your database, directory, or other datastore. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** tab. For more information, see Defining attribute sources and user lookup.
Expression (if enabled)	Select **Expression** to support complex mapping requirements; for example, transforming incoming values into different formats. Additionally, the HTTP request is retrieved as a Java object instead of text. For this reason, select **Expression** as the source and use OGNL expressions to evaluate and return specific information from the HTTP request. Applicable only if you have enabled the use of expressions in PingFederate. For more information, see Attribute mapping expressions.
No Mapping	Select **No Mapping** to ignore the **Value** field.

Source	Description
Text	Select **Text** to return the value you enter under **Value**. There are a variety of reasons to use a static text value. For example, if the target web application provides a service based on the name of your organization, you can provide the attribute value as a constant. You can mix text with references to attributes from the IdP adapter contract by using the `$\{attribute}` syntax. You can also enter references to attributes from configured attribute sources by using the `$\{ds.attr-source-id.attribute}` syntax, where `attr-source-id` is the **Attribute Source ID** value you entered on **Attribute Sources & User Lookup > Data Store** and `attribute` is an attribute from the datastore. For more information, see Defining attribute sources and user lookup. **Tip** You can reference attribute values in the form of `$\{attributeName:-defaultValue}` . The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

6. Repeat these steps until all attributes are configured.

7. Click **Done**.

Defining issuance criteria for IdP adapter contract

You can manage criteria that PingFederate evaluates to determine whether to issue an identity provider (IdP) adapter contract token for a user.

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Authentication > Integration > IdP Adapters**.
2. Click the name of the existing instance you want to change in the **Instance Name** list.
3. Click **Adapter Contract Mapping > Configure Adapter Contract > Issuance Criteria**.
4. In the **Source** list, select the attribute's source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

+

Source	Description
Adapter	Select to evaluate attributes from the IdP adapter instance.
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

+ . In the **Attribute Name** list, select the attribute to be evaluated.

+

Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**

- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value …** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+

To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

+ . **Optional:** Repeat to add more criteria.

+ . If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

+ For more information, see Attribute mapping expressions.

1. Click **Show Advanced Criteria**.
2. In the **Expression** field, enter the required expressions.
3. In the **Error Result** field, enter an error code or message.

> **Note**
>
> If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click **Add**.
5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.

6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing an IdP adapter contract

On the **Create Adapter Instance** window's **Summary** tab, review the adapter contract.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Reviewing and saving an IdP adapter configuration

Review your identity provider (IdP) configuration and save your changes on the **Summary** tab.

Steps

- To keep your changes, click **Save**.
- To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
- To discard your changes, click **Cancel**.

Authentication applications and the authentication API

The PingFederate authentication API is a JSON-based API that enables end-user interactions, such as credential prompts, to be handled by an external web application. This API does so by providing access to the current state of the flow as an end user steps through a PingFederate authentication policy.

Authentication flows are initiated through browser-based single sign-on (SSO) application endpoints, such as `/idp/startSSO.ping`, or a protocol request, such as an OpenID Connect authentication request received at the authorization endpoint: `/as/authorization.oauth2`. As PingFederate runs the configured authentication policy, if it encounters an API-capable adapter or selector, and an authentication application is configured for the policy, PingFederate redirects to the authentication application's URL, passing the ID of the flow in the `flowId` query parameter.

The authentication application can then retrieve the current state of the flow by issuing a `GET` request to the `/pf-ws/authn/flows/{flowId}` endpoint. The `_links` field in the response lists the available authentication actions that can be performed from the current state. To invoke an action, the authentication application sends a `POST` request to the `/pf-ws/authn/flows/{flowId}` endpoint.

The API client can choose authentication actions using either an `action` query parameter or a custom content type. The `action` query parameter should be used when your networks block custom content types, and is specified using the following format:

```
/pf-ws/authn/flows/<flowId>?action=<actionID>
```

For example, to check username and password, pass the `checkUsernamePassword` action ID through the query parameter as follows:

```
/pf-ws/authn/flows/<flowId>?action=checkUsernamePassword
```

For the custom content type, the action ID is specified using the Content-Type HTTP request header in the following format:

```
application/vnd.pingidentity.<actionId>+json
```

For example, to indicate a username and password check from the HTML Form Adapter, specify the following:

```
application/vnd.pingidentity.checkUsernamePassword+json
```

When the application invokes an action, PingFederate responds either with the next state for the flow or an error.

When the user completes the authentication policy steps successfully, the authentication API returns a `RESUME` status to the authentication application. This status indicates that the API client should redirect the user's browser to the `resumeUrl` specified in the response. PingFederate will then be responsible for the final step in the flow, such as passing a SAML assertion to a partner. A `RESUME` status will also be sent if PingFederate encounters an identity provider (IdP) connection in the policy tree, or an IdP adapter or selector that is not API-capable. When the API client redirects the user, PingFederate will take the steps needed to invoke the authentication source.

If the user has interacted with an authentication application and the flow terminates with an error, the API client will receive a `FAILED` status from the API.

Authentication applications must be highly trusted.

> **Note**
>
> To avoid issues with third-party cookies in some browsers, give the authentication application the same parent domain as the PingFederate base URL.

Key concepts

Flow

The SSO transaction invoking the authentication API.

States

The available states (if any) for a given API-enabled adapter or selector.

Current state

Indicates what the adapter or selector is ready to do next.

Actions

The available actions (if any) for a given state.

Default authentication application

In addition to specifying an authentication application for each policy, you can also configure a default application. PingFederate uses the default application if it encounters an API-capable adapter outside of a policy tree. For example, the default application activates if the user navigates to the Change Password endpoint, `/ext/pwdchange/Identify`, or the Account Recovery endpoint, `/ext/pwdreset/Identify`. PingFederate also uses the default application if authentication policies are disabled, or if the flow falls through to a default authentication source.

Managing authentication applications

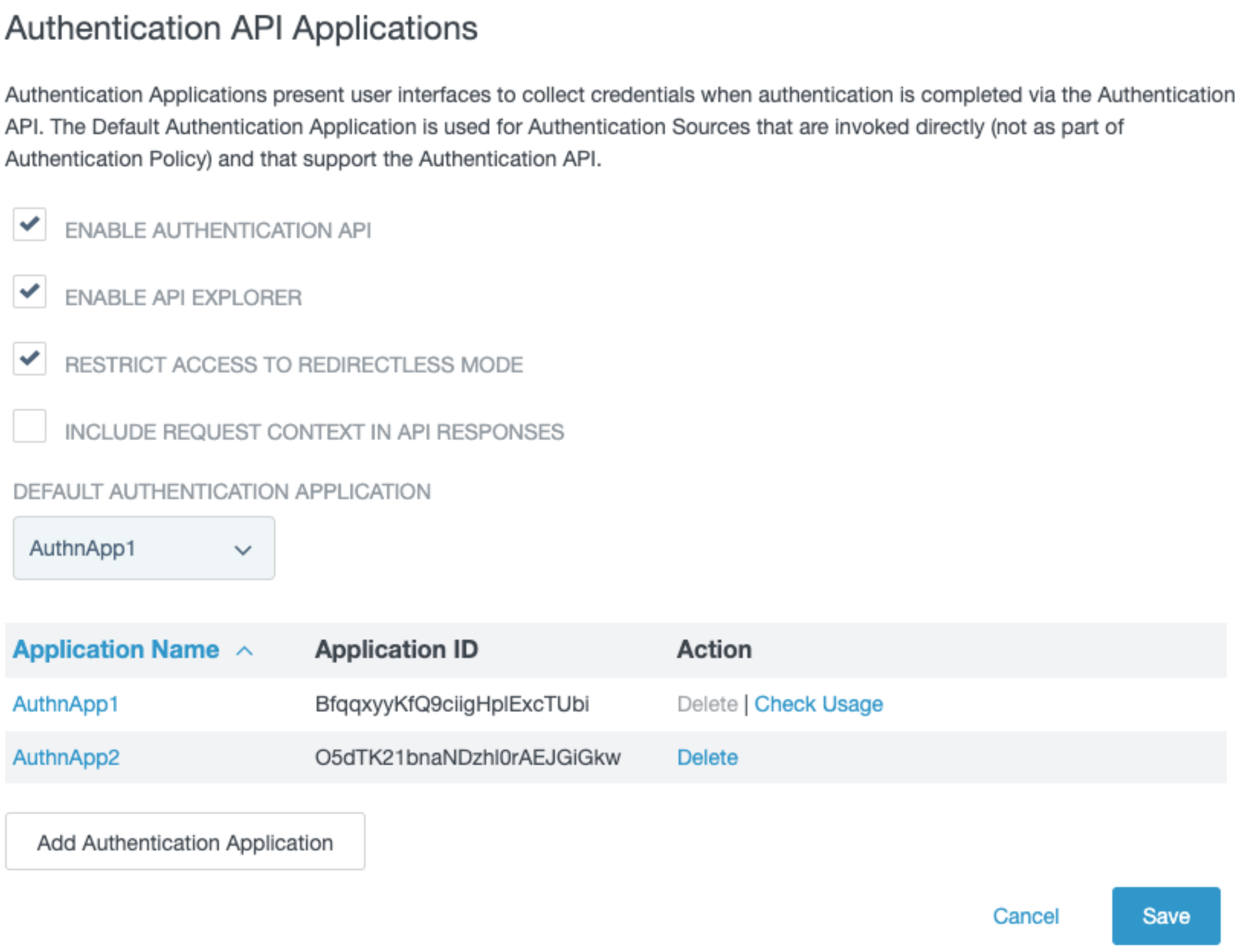
You can create and manage authentication applications that use the authentication API.

About this task

Authentication applications display user interfaces to collect credentials when authentication is completed through the PingFederate authentication API. The default authentication application is used for authentication sources that support the authentication API functionality and are invoked directly, rather than as part of an authentication policy.

Steps

1. To manage authentication applications, go to **Authentication** > **Integration** > **Authentication API Applications**.



Authentication API Applications

Authentication Applications present user interfaces to collect credentials when authentication is completed via the Authentication API. The Default Authentication Application is used for Authentication Sources that are invoked directly (not as part of Authentication Policy) and that support the Authentication API.

☑ ENABLE AUTHENTICATION API

☑ ENABLE API EXPLORER

☑ RESTRICT ACCESS TO REDIRECTLESS MODE

□ INCLUDE REQUEST CONTEXT IN API RESPONSES

DEFAULT AUTHENTICATION APPLICATION

AuthnApp1

Application Name	Application ID	Action
AuthnApp1	BfqqxyyKfQ9ciigHplExcTUbi	Delete | Check Usage
AuthnApp2	O5dTK21bnaNDzhl0rAEJGiGkw	Delete

2. To toggle the availability of authentication API support, select or clear the **Enable Authentication API** check box.

> **Note**
>
> The **Enable API Explorer**, **Restrict Access to Redirectless Mode**, and **Include Request Context in API Responses** check boxes are applicable and shown if the **Enable Authentication API check** box is selected.

Option	Description
Enable API Explorer	PingFederate includes an API Explorer that allows you to view the states, actions, and models available for the vari capable adapters and selectors included in your PingFederate environment. The endpoint for the Authentication API Explorer is `/pf-ws/authn/explorer`. For more information, see Exploring authentication API. This check box is enabled by default.

Option	Description
Restrict Access to Redirectless Mode	It is strongly recommended to enable the **Restrict Access to Redirectless Mode** setting. If it is not enabled, auther applications can use a user's existing session to obtain tokens for any public client defined in the deployment, that client with no authentication method defined. Enabling **Restrict Access to Redirectless Mode** ensures that authentication applications can only obtain tokens fo client specified in the application's settings. When you enable this setting, make sure to update authentication app that use redirectless mode and specify the client that they are allowed to use. For more information on how to allow highly-trusted authentication applications to employ the PingFederate Authentication API, see Configuring authentication applications. **Restrict Access to Redirectless Mode** is enabled by default.
Include Request Context in API Responses	To pass single sign-on (SSO) request context parameters and tracked parameters to authentication applications, s **Include Request Context in API Responses** check box. Enabling this feature allows authentication API clients to use the context of SSO requests to make decisions and ch branding. When enabled, the authentication API response includes the `requestContext` parameter of type Map. T following parameters are included when they are relevant to the SSO transaction: **`pluginId`** The ID of the identity provider (IdP) adapter or the authentication selector **`entityId`** The ID of the service provider (SP) connection used in the SSO transaction **`applicationName`** The name of the SP connection or OAuth client used in the SSO transaction **`client_id`** The ID of the OAuth client used in the SSO transaction **`spAdapterId`** The ID of the SP adapter used in the SSO transaction **`oidcUiLocales`** The OIDC `ui_locales` **`trackedHttpParams`** An array of the tracked HTTP parameters passed when processing authentication policies **`extendedProperties`** Passes defined extended properties to all applicable velocity templates and as a request context paramete authentication API **Note** Except for tracked HTTP parameters, these parameters do not include sensitive information. Whether tracke parameters include sensitive information depends on which parameters you choose to track in policies. For information about configuring tracked HTTP parameters, see Defining authentication policies.

3. In the **Default Authentication Application** section, perform any of the following actions.

Option	Action
Default Authentication Application	Select an application from the list to designate as the default authentication application.

Option	Action
Check Usage	Click to open a pop-up window listing the configurations in which the authentication is used. ⓘ **Note** This is only available for the default authentication application.
Add Authentication Application	Click to add a new authentication application. See Configuring authentication applications.
Delete	Click to remove an authentication application.

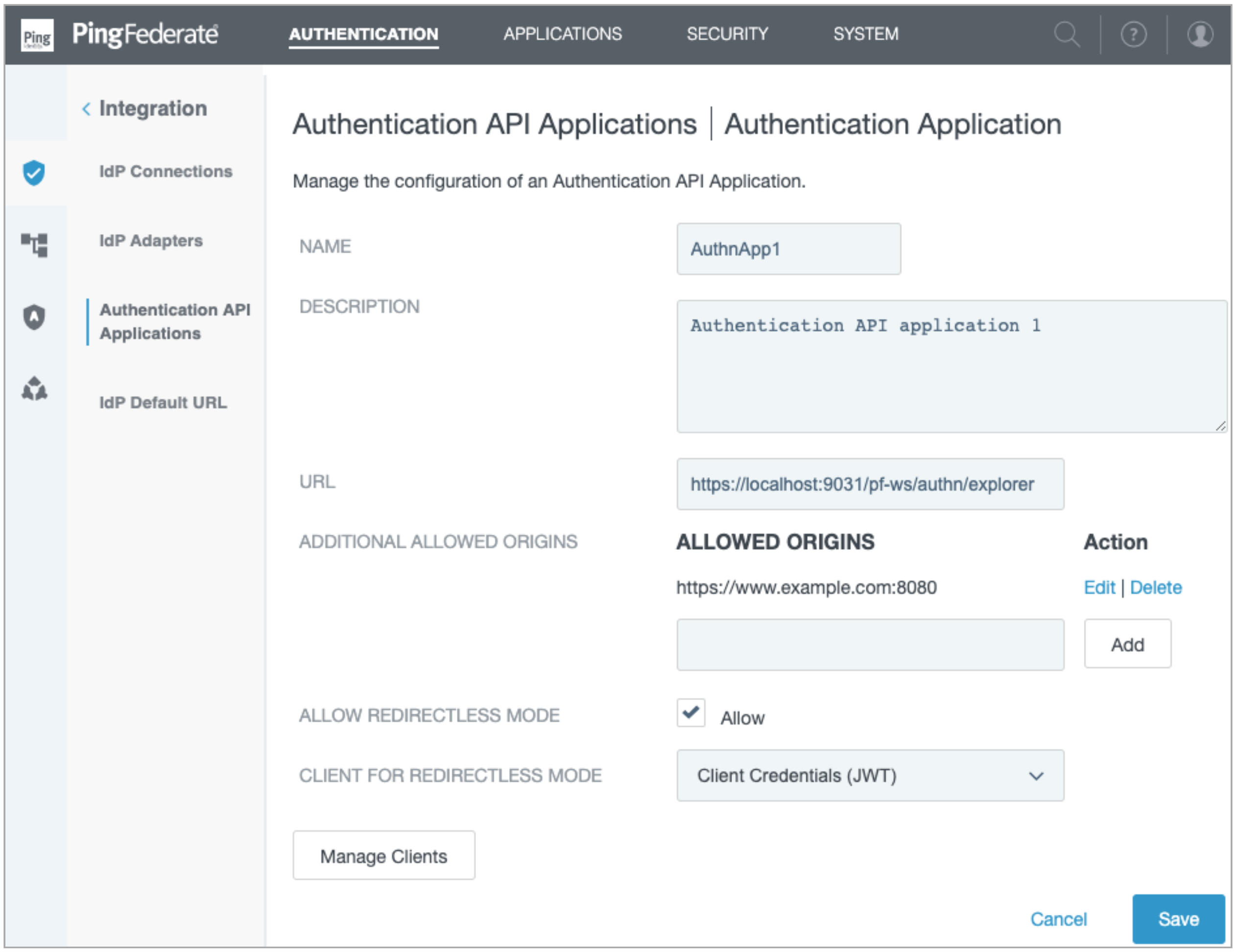
4. Click **Save**.

Configuring authentication applications

You can let highly-trusted authentication applications employ the PingFederate Authentication API.

About this task

The following procedure describes how to use the **Authentication Application** window to integrate an authentication application with PingFederate.



Authentication API Applications | Authentication Application

Manage the configuration of an Authentication API Application.

NAME: AuthnApp1

DESCRIPTION: Authentication API application 1

URL: https://localhost:9031/pf-ws/authn/explorer

ADDITIONAL ALLOWED ORIGINS

ALLOWED ORIGINS	Action
https://www.example.com:8080	Edit | Delete

ALLOW REDIRECTLESS MODE: ☑ Allow

CLIENT FOR REDIRECTLESS MODE: Client Credentials (JWT)

Steps

1. Go to **Authentication > Integration > Authentication API Applications**.

   Choose from:

   - To integrate a new application, click **Add Authentication Application**.
   - To modify an existing application's settings, click the **Application Name**.

2. Provide information for each field.

   For more information, see the following table.

Field	Description
Name	The name of the authentication application.
Description	An optional description of the authentication application.
URL	The URL of the authentication application.

Field	Description

Additional Allowed Origins	Enter any number of trusted origins to enable cross-origin resource sharing (CORS) support for the authentication API endpoint.
	Once configured, client-side web applications from the trusted origins are allowed to make requests to the PingFederate authentication API endpoint. For more information about CORS, see W3C's recommendation on Cross-Origin Resource Sharing.
	The following list includes example origin formats and their expected behaviors:
	`https://www.example.com`
	CORS requests originating from https://www.example.com are allowed.
	`https://www.example.com:8080`
	CORS requests originating from https://www.example.com:8080 are allowed.
	`https://www.example.com:`
	CORS requests originating from https://www.example.com:<*any port*> are allowed.
	Note
	Given this sample entry, a port number is required in the `Origin` request header.
	Important
	Although using the wildcard character provides the convenience of allowing multiple origins with one entry, we recommend adding individual origins to limit CORS requests to a list of trusted hosts.
	Click **Add** to save an entered origin. Repeat to add multiple origins.

Allow Redirectless Mode	To allow the authentication application to use redirectless mode with a specified OAuth client, select this check box. Then select a **Client for Redirectless Mode**. This check box is not selected by default. It is visible only if the **Restrict Access to Redirectless Mode** check box on the **Authentication API Applications** window is selected. For more information, see Managing authentication applications.
Client for Redirectless Mode	Select the OAuth client that will use this authentication application in redirectless mode. The client must allow authentication API redirectless mode. For more information, see Configuring OAuth clients. This field is visible only if the **Allow Redirectless Mode** check box is selected.

3. To keep your configuration, click **Save** to or click **Cancel** to discard any changes made.

Configuring a default URL and error message

As an identity provider (IdP), you can optionally prompt end users to confirm their single logout (SLO) requests and provide a default URL indicating a successful SLO to the end-user, if no other page is designated.

About this task

You can also customize an error message to be displayed as part of the error page rendered in the end-user's browser if an error occurs during IdP-initiated single sign-on (SSO). For example, you might consider modifying the default text to include useful information regarding whom the user should contact or what their next step should be.

Your application or your partner's application can supply the SLO URL at runtime. However, if none is provided, PingFederate will use the default value you enter on this window. For more information, see IdP endpoints.

If you leave the default URL blank, PingFederate provides a built-in landing page for the user. This web page is among the templates you can modify with your own branding or other information. For more information, see Customizable user-facing pages).

Steps

1. Go to **Authentication > Integration > IdP Default URL**.
2. Select the check box to prompt the user to confirm SLO.
3. Enter a default URL to send the user to on successful SLO.
4. Enter a custom error message to display on unsuccessful SLO.

> **Note**
>
> The error message is displayed only when the application calling the start-SSO endpoint does not explicitly provide its own error page URL. The default entry in this field is used to localize the message. For information about how to use the PingFederate localization feature, see Localizing messages for end users. If localization is not needed, you can also specify a default message in this field.

5. Click **Save** to save your changes.

Viewing IdP application endpoints

Web-application developers at your site need to know the application endpoints to initiate transactions through PingFederate.

Steps

- Go to **System > Endpoints > IdP Application Endpoints** to see a list of endpoints and descriptions applicable to your federation role.

  These endpoints are built into PingFederate and cannot be changed.

  For specific parameters required or allowed for these endpoints, see IdP endpoints and System-services endpoints.

IdP protocol endpoints

PingFederate provides a list of identity provider (IdP) protocol endpoints and exportable metadata for your configuration.

You can find a list of applicable SAML, WS-Federation, and WS-Trust STS endpoints in **System > Endpoints > IdP Endpoints**. The pop-up window displays only those endpoints related to the federation protocols enabled on **System > Server > Protocol Settings > Federation Info**. These endpoints are built into PingFederate and cannot be changed.

Your federation partners or security token service (STS) clients need to know the applicable IdP services endpoints to communicate with your PingFederate server. Configured service endpoints for SAML connections are included in metadata export files.

PingFederate provides a favicon for all protocol endpoints. For more information, see Customizing the favicon for application and protocol endpoints.

The following table describes each endpoint.

Service	URL and Description
Single Logout Service (SAML 2.0)	`/idp/SLO.saml2` The URL that receives and processes logout requests and responses.
Single Sign-on Service (SAML 2.0)	`/idp/SSO.saml2` The SAML 2.0 implementation URL that receives authentication requests for processing.
Artifact Resolution Service (SAML 2.0)	`/idp/ARS.ssaml2` The SOAP endpoint that processes artifacts returned from a federation partner to retrieve the referenced XML message on the back channel. See the note at the end of this table.
Attribute Query Service (SAML 2.0)	`/idp/attrsvc.ssaml2` The SAML implementation that receives and processes attribute requests. See the note at the end of this table.
Single Sign-on Service (SAML 1.x)	`/idp/isx.saml1` The SAML 1.x implementation of IdP intersite transfer service (ISX) to which clients are redirected for single sign-on (SSO) requests.
Artifact Resolution Service (SAML 1.x)	`/idp/soap.ssaml1` The SOAP endpoint that processes artifacts returned from a federation partner to retrieve the referenced XML message on the back channel. See the note at the end of this table.
Single Sign-on Service (WS-Federation)	`/idp/prp.wsf` The WS-Federation implementation URL that receives and processes security-token requests and single log-out (SLO) messages.

Service	URL and Description

WS-Trust STS (two endpoints)	`/idp/sts.wst`
	The SOAP endpoint that receives and processes security-token requests from STS clients (web service clients at the IdP site) to be exchanged for a SAML token based on the configured service provider (SP) connection.
	`/pf/sts.wst`
	Initiates direct STS token-to-token exchange and token validation from an IdP token processor to an SP token generator, when that feature is configured. For more information, see Token translator mappings.
	ⓘ **Note**
	If multiple token-processor instances of the same type are configured for the same connection or token-to-token mapping, a query parameter, `TokenProcessorId`, must be added to either of these endpoints. For more information, see Managing token processors.
	See the note at the end of this table.

Important If mutual SSL/TLS is used for authentication, a secondary PingFederate listening port must be configured and used by partners or STS clients for the relevant endpoints— `.ssaml` and `*.wst` For more information, see Configuring PingFederate properties.	

Virtual server ID support

For SAML connections using multiple virtual server IDs, each virtual server ID has its own set of protocol endpoints. For more information, see Multiple virtual server IDs. You can export connection metadata for your partner from **System > Protocol Metadata > Metadata Export**. For more information, see Exporting connection-specific SAML metadata.

For WS-Federation (and SAML) connections using multiple virtual server IDs, you can provide your partner the federation metadata endpoint, `/pf/federation_metadata.ping`, with the `PartnerSpId` and `vsid` parameters, as in the following example.

Partner's entity ID	Your virtual server ID	Federation metadata URL
SP	idev1	https://www.example.com/pf/federation_metadata.ping?PartnerSpId=SP&vsid=idev1
	idev2	https://www.example.com/pf/federation_metadata.ping?PartnerSpId=SP&vsid=idev2

In this example, the base URL and the runtime port of your PingFederate server are www.example.com and 443, respectively.

When the request includes the `vsid` parameter, the federation metadata endpoint returns information that is specific for a given virtual server ID.

For WS-Trust STS, you can provide your partner the STS metadata endpoint `/pf/sts_mex.ping` with the `PartnerSpId` and `vsid` parameters. When the STS metadata request includes the `vsid` parameter, the STS metadata endpoint returns information that is specific for a given virtual server ID.

For more information about these metadata endpoints, see System-services endpoints.

> **Note**
>
> The virtual server ID concept does not apply to the `/pf/sts.wst` endpoint because token-to-token exchange does not involves any connections. As needed, you can pass the token-to-token endpoint to your partners as-is.

SP connection management

As an identity provider (IdP), you manage connection settings to support the exchange of federation-protocol messages such as SAML, WS-Federation, or WS-Trust with a service provider (SP) or security token service (STS) client application at your site.

The settings your IdP manages include:

- User attributes that you expect to send in a single sign-on (SSO) token, including SAML assertions, WS-Trust STS SAML tokens, or WS-Federation JSON Web Tokens (JWT).
- User attributes that are sent using the SAML Attribute Query profile, if that profile is used. For more information, see Configuring the Attribute Query profile in an SP connection.
- The protocol, profiles, and bindings of the connection, including detailed security specifications such as the use of back-channel authentication, digital signatures, signature verification, and XML encryption

To establish a connection, you and your partner must decide this information in advance. For more information, see Federation planning checklist.

If your agreement includes sending assertions containing attribute values from local datastores, you must define the required datastores. For more information, see Datastores.

> **Note**
>
> If you delete a provisioning connection, the users are not deleted from the SP. They are simply no longer managed by PingFederate.

Administrative interface

Manage connection settings using the **SP Affiliations** window, accessed from **System > Protocol Metadata**, which organizes the settings into a series of primary tasks.

Some primary tasks have one or more levels of sub task. Each primary or sub task has its own tab for managing one or more settings. You can move to a sibling task using the **Next** or **Previous** button. If you are on a sub task, you can also move to its parent task using the **Done** button.

When creating a new connection, you can save your progress using the **Save Draft** button. Not all windows offer this option. When you reach the **Activation & Summary** tab, you must click **Save** to complete the new connection.

When editing an existing connection, make changes and then click **Save** to commit your changes. You are not required to step through all window to reach the **Activation & Summary** window before you can save your changes.

> **Note**
>
> The **Save** button is available on most tabs. If a window does not show a **Save** button, click **Next** or **Done** until you reach a window where you can use its **Save** button to commit your changes.

Accessing SP connections

The **SP Connections** window lists all service provider connections and displays up to 20 connections at a time. As needed, you can sort connections by connection name, partner connection ID, default virtual server ID, creation date, or last modified timestamps; narrow by protocol type or status; use the pagination controls to navigate through your connections; and search for connections by name or ID.

About this task

A connection is included in the search results so long as its name or ID is a partial, case-insensitive match to a search term.

Steps

- Go to **Applications > Integration > SP Connections**.
- To edit a connection, select the connection by its name. For the setting you want to make a change, select the corresponding window title and then follow the configuration wizard to complete the task.
- To create a connection, click **Create Connection** and follow the on-screen steps.
- To copy a connection, click **Select Action > Copy** and then follow the on-screen steps.

  This is most useful if the new connection and the source connection share many common setting values.

  > **Note**
  >
  > PingFederate doesn’t include outbound provisioning configurations in connection copies. For more information, see Configuring outbound provisioning.

- To export a connection, click **Select Action > Export Connection** and then save the XML file as prompted.

  This is useful in situations where you want to make a backup of a connection prior to changing it.

- To import a connection, click **Import Connection**. For more information, see Importing a connection.

  If the connection already exists, you have the option to overwrite the existing connection.

  > **Note**
  >
  > Prior to the import, you can modify the XML file to suit your needs. The XML file can also be imported to another PingFederate environment acting in the same federation role (IdP) at your site. The source and the target must run the same version of PingFederate.

- To export metadata for any SAML Browser SSO connection, click **Select Action > Export Metadata** and then follow the on-screen instructions.
- To update a SAML Browser SSO connection, click **Select Action > Update with Metadata**, then follow the on-screen instructions. For more information, see Importing SP metadata.

You can update a connection via a metadata XML file or a metadata URL.

> **Important**
>
> The update operation might require additional configuration. Review the connection after the update operation.

- Click the toggle to enable or disable a connection.
- To remove a connection, click **Select Action** > **Delete**.
- To override the verbosity of runtime transaction logging for all SP connections, click **Show Advanced Fields** and the select the desired override option.

Override option	Description
Off	Select this option and let the per-connection **Logging Mode** configuration determine the amount of information PingFederate records in the runtime transaction log. This is the default selection.
On	Select this option, followed by one of the four logging modes, to set the verbosity of runtime transaction logging for all SP connections. This is most useful when troubleshooting an issue that affects multiple connections.

- To turn off automatic multi-connection error checking, click **Show Advanced Fields** > **Disable Automatic Connection Validation** check box.

  This check box is not selected by default.

  Once selected or cleared, the state of this setting is also reflected on the **Authentication** > **Integration** > **IdP Connections** window.

  For more information about this advanced setting and its impact, see Configuring automatic connection validation.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Resolving SP connection errors

You can diagnose and resolve service provider (SP) connection errors from the **SP Connections** menu.

About this task

PingFederate automatically validates configured connections before displaying them on the **SP Connections** window. This validation ensures these connections have not been adversely affected by any subsequent changes in the supporting components, such as an adapter instance or an authentication policy contract.

If errors are found, the administrative console displays a visual cue next to the applicable connections.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.

2. To resolve the error, select the connection and follow the on-screen instructions to modify the configuration, one connection at a time.

Importing a connection

You can import a connection in the **Import Connection** window.

About this task

> **Note**
>
> Prior to the import, you can modify the XML file to suit your needs. The XML file can also be imported to another PingFederate environment acting in the same federation role. The source and the target must run the same version of PingFederate.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.
2. Click **Import Connection**.
3. Click **Choose File** and browse to a connection XML file.
4. Select the **Allow Update** check box to overwrite an existing connection with the imported file.
5. Click **Import** and **Done**.

Updating a SAML connection using metadata

You can update an existing SAML connection using a metadata file or a metadata URL from your partner.

About this task

This manual update is independent from the optional and per-connection automatic update feature.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.
2. For the SAML connection that you want to update, click **Select Action** > **Update with Metadata**.
3. See the following steps to import or update metadata. Instructions vary depending on the medium of the metadata.

Metadata medium	Steps
Metadata XML File	1. On the **Import Metadata** window, select the **File** option. 2. Choose the metadata file, and then click **Next**. **Note** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata verification certificate on the **Import Certificate** tab, and then click **Next**. 3. On the **Metadata Summary** window, review the signature information to evaluate the authenticity of the metadata. 4. Click **Save**.
A metadata URL	1. On the **Import Metadata** window, select the **URL** option. 2. Select the metadata from the **Metadata URL** list. **Tip** If the metadata you want is not shown in the list, click **Manage Partner Metadata URLs** and enter the URL in the text box. 3. Click **Load Metadata**. **Note** If there is a digital signature error, click **Manage Partner Metadata URLs** to resolve the issue. 4. Click **Save**.

4. On the **Connections** window, click **Save**.

Result

> **Note**
>
> If the endpoints in the metadata share the same base URL (protocol, hostname, and port), PingFederate uses this information to populate the Base URL field. Consequently, individual endpoints on other windows do not include this information. Only relative paths are shown.

Example

A service provider (SP) has just changed its signing certificate and published a new metadata with the new certificate. To minimize the impacts to your users, you as the identity provider (IdP) can update the SP connection using the metadata immediately.

1. Access the **SP Connections**window from **Applications > Integration > SP Connections**.

2. For the applicable SAML connection, click **Select Action** > **Update with Metadata**.
3. Follow the workflow to complete the task.

Choosing an SP connection template

The **Connection Template** tab allows you to choose a quick-connection template for new connection if your installation includes an optional PingFederate SaaS Connector.

About this task

> **Tip**
>
> When you select a connection template, many connection settings are configured for you automatically.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.
2. Click **Create Connection**.
3. To use a template, select **Use a template for this connection**, then choose the template and enter additional information as required.

   > **Important**
   >
   > After you click **Next**, you cannot return to this window and make a different selection. If you intended to use a different template or no template, you must create a new connection.

Choosing an SP connection type

You can manually create service provider (SP) connections in PingFederate using browser single sign-on (SSO), WS-Trust security token service (STS), outbound provisioning, or any combination thereof.

About this task

If you are not using a connection template, which pre-configures browser-based SSO, indicate on the **Connection Type** tab whether the connection to this partner is for Browser SSO, WS-Trust STS, outbound provisioning, or any combination of them.

> **Tip**
>
> You can add STS, OAuth, and outbound provisioning support to any existing SSO connection, or vice versa, at any time.

> **Note**
>
> If your partner's deployment supports multiple protocols and you intend to communicate using more than one, you must set up a separate connection for each protocol. Each connection must use a unique (partner) connection ID.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.

2. Click **Create Connection**.

3. Select **Do not use a template for this connection**.

4. To configure a connection for secure browser-based SSO, select the **Browser SSO Profiles** check box.

   If you are not using a connection template, you must select the applicable protocol from the list when establishing a new connection.

   For a WS-Federation connection, select the desired token type, either **SAML 1.1**, **SAML 2.0**, or **JWT** (JSON Web Token).

   > **Note**
   >
   > For information about creating a SAML application, see Configuring a SAML application in PingFederate.

   > **Tip**
   >
   > If you are creating a WS-Federation connection to Microsoft Windows Azure Pack, select JWT as the token type.

   > **Tip**
   >
   > PingFederate can encrypt the subject and attributes of SAML 2.0 assertions.
   > For information about configuring encryption policies on a PingFederate identity provider (IdP), see Configuring XML encryption policy (SAML 2.0).
   > For information about configuring encryption policies on a PingFederate SP, see Specifying XML encryption policy (for SAML 2.0).

5. **Optional:** Choose one or both of the following depending on your configuration needs.

Connection Template	Step
WS-TRUST STS	Select the **WS-Trust STS** check box.
Outbound Provisioning	Select **Outbound Provisioning** and then select the provisioning type from the list.

6. If your PingFederate license manages connections by groups, select a license group for this connection.

   This option is not shown for unrestricted or other types of licenses.

7. To save your settings, click **Next**.

Choosing SP connection options

On the **Connection Options** tab, you can enable browser-based single sign-on (SSO), Attribute Query, or both for the current connection.

Before you begin

For initial steps in creating a service provider (SP) connection, see Choosing an SP connection type.

Steps

1. Choose one or more of the following options.

Connection option	Description
Browser SSO	Select to create a connection for browser-based SSO.
IdP Discovery	Select to enable identity provider (IdP) discovery. This option is only available if you have configured IdP discovery. For more information, see Configuring standard IdP Discovery.
Attribute Query	Select to create a connection that facilitates the SAML 2.0 Attribute Query profile. For more information, see Attribute Query and XASP.

2. To save your changes, click **Next**.

Importing SP metadata

When creating or modifying service provider (SP) connections, PingFederate allows you to import metadata from an XML file or a metadata URL.

About this task

If you are using one of the SAML protocols without a connection template, you can expedite the setup by one of the following actions:

- Import a metadata file
- Select a metadata URL

When you select a metadata URL, PingFederate also enables the automatic update option and checks the metadata periodically. If PingFederate detects changes in the partner's signing certificates for digital signature verification, encryption key, or contact information, it updates the connection automatically. For better housekeeping, the update process removes verification certificates from the connection when the partner no longer maintains them in its metadata. In a clustered environment, PingFederate automatically replicates verification certificates and encryption key changes to all engine nodes. Offline engine nodes will also consume these changes as they restart and rejoin the cluster. If you prefer to update the connection manually, you can clear the **Enable Automatic Reloading** check box.

You can configure reload frequency at **System** > **Protocol Metadata** > **Metadata Settings** > **Metadata Lifetime** tab. The default reload frequency is daily.

We recommend you turn on notifications for SAML metadata update events at **System** > **Monitoring & Notifications** > **Runtime Notifications**.

Note

The notification message provides a list of the applicable items if the metadata contains changes that require additional configuration.

After creating the connection, you can add, remove, or change the metadata URL associated with the connection in the **Metadata URL** tab. In addition, you can toggle the **Enable Automatic Reloading** check box for the connection.

> **Tip**
>
> Using a metadata URL with automatic reloading streamlines the configuration process. For example, you can quickly establish a browser SSO connection to an InCommon-participating partner. For more information, see www.incommon.org/participants.

Steps

1. Select from one of the following steps to import or update metadata.

Metadata medium	Steps
Metadata file	1. On the **Import Metadata** tab, select the **File** option. 2. Choose the metadata file, and then click **Next**. **Note** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata verification certificate on the **Import Certificate** tab, and then click **Next**. 3. On the **Metadata Summary** tab, review the signature information to evaluate the authenticity of the metadata.

Metadata medium	Steps
Metadata URL	1. On the **Import Metadata** tab, select the **URL** option. 2. Select the metadata from the **Metadata URL** list. **Tip** If the metadata you want is not shown in the list, click **Manage Partner Metadata URLs**. For more information, see Manage Partner metadata URLs. 3. Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. **Note** A warning will display if you do not have runtime notifications enabled. To enable these notifications, go to **System > Monitoring & Notifications > Runtime Notifications** and select the **Notification for SAML Metadata Update Events** box. 4. Click **Load Metadata**. **Note** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note** If there is a digital signature error, click **Manage Partner Metadata URLs** to resolve the issue.

2. Click **Next**.

Identifying the SP

On the **General Info** tab, you provide your partner's unique federation identifier, the display name of the connection, and some other optional information, such as virtual server IDs, contact information, and logging mode for runtime transaction logging.

Steps

1. For information on initial steps for managing SP connections, see Choosing an SP connection type.
2. Provide the basic information to identify your partner.

   See the following table for more information.

Field	Description
Partner's Entity ID, Issuer, Partner's Realm, or Connection ID (Required)	The published, protocol-dependent, unique identifier of your partner. For a SAML 2.0 connection, this is your partner's SAML Entity ID. For a SAML 1.x connection, this is the `Audience` your partner advertises. This ID may have been obtained out-of-band or using a SAML metadata file. For a WS-Federation connection, this is your partner's Realm. For a security token service (STS)-only connection, you can designate any unique identifier.
Connection Name (Required)	A plain-language identifier for the connection. For example, a company or department name. This name is displayed in the connection list on the administrative console.
Virtual Server IDs	If you want to identify your server to this connection partner using an ID other than the one you specified at **System** > **Server** > **Protocol Settings** > **Federation Info**, enter a virtual server ID in this field and click **Add**. Enter additional virtual server IDs as needed.
Base URL	The fully qualified host name and port on which your partner's federation deployment runs. For example, *https://www.example.com:9031*. This entry is an optional convenience, allowing you to enter relative paths to specific endpoints, instead of full URLs, during the configuration process.
Company	The name of the partner company to which you are connecting.
Contact Name	The contact person at the partner company.
Contact Number	The phone number of the contact person at the partner company.
Contact Email	The email address for the contact person at the partner company.
Application Name	The name of the application, accessible through the IdP Adapter interface `IdpAuthenticationAdapterV2` in the PingFederate Java SDK. This field is not applicable to an STS-only connection.
Application Icon URL	The URL of the application icon, accessible through the IdP Adapter interface `IdpAuthenticationAdapterV2` in the PingFederate Java SDK. Note that this field is not applicable to an STS-only connection.
Logging Mode	The level of transaction logging applicable for this connection.

3. After entering the relevant identification information, click **Next**.

Populating extended property values for SP connections

Add, modify, or delete extended properties for service provider (SP) connections on the **Extended Properties** page.

About this task

Extended property values can serve as metadata. They can also help drive authentication requirements. For more information, see Extended properties.

Steps

1. Go to **System** > **Server** > **Extended Properties**.
2. Enter the **Name** and **Description** for the property you want to add.

   Any extended properties that are defined will be passed to all applicable velocity templates and as a request context parameter in the authentication API.
3. **Optional:** Select the **Multivalued** check box to indicate that the property permits multiple values.
4. Click **Add**.

Configure IdP Browser SSO

Browser-based single sign-on (SSO), also known as Browser SSO, relies on a user's web browser and HTTP requests to broker identity-federation messaging in XML or JSON web token (JWT) between an identity provider (IdP) and a service provider (SP).

Go to **Applications** > **Integration** > **SP Connections** to access an existing or create a new SP connection. For more information, see Accessing SP connections.

From the **Browser SSO** tab inside your SP connection instance, click **Configure Browser SSO** and follow the steps below based on your federation protocol.

> **Tip**
>
> Many steps involved in setting up a federation connection are protocol-independent. They are required steps for all connections, regardless of the associated standards. For more information, see Federation roles.
> Some steps are required under the applicable protocol, while others are optional. Still others are required only based on certain selections. The administrative console determines the required and optional steps based on the protocol and dynamically presents additional requirements or options based on selections.
> The following sections provide sequential information about every step you might encounter while configuring browser-based SSO, depending on the protocol you are using for a particular connection.

SAML 2.0 configuration steps

- Choosing SAML 2.0 profiles
- Setting an SSO token lifetime
- Configuring SSO token creation
  - Choosing an identity mapping method for IdP SSO
  - Setting up an attribute contract
  - Managing authentication source mappings
- Configuring protocol settings
  - Setting Assertion Consumer Service URLs (SAML)

    - Specifying SLO service URLs (SAML 2.0)
    - Choosing allowable SAML bindings (SAML 2.0)
    - Setting an artifact lifetime (SAML)
    - Specifying artifact resolver locations (SAML 2.0)
    - Defining signature policy (SAML)
    - Configuring XML encryption policy (SAML 2.0)

SAML 1.x configuration steps

- Setting an SSO token lifetime
- Configuring SSO token creation
    - Choosing an identity mapping method for IdP SSO
    - Setting up an attribute contract
    - Managing authentication source mappings
- Configuring protocol settings
    - Setting Assertion Consumer Service URLs (SAML)
    - Setting a default target URL (SAML 1.x)
    - Setting an artifact lifetime (SAML)
    - Defining signature policy (SAML)

WS-Federation configuration steps

- Setting an SSO token lifetime
- Configuring SSO token creation
    - Choosing an identity mapping method for IdP SSO
    - Setting up an attribute contract
    - Managing authentication source mappings
- Configuring protocol settings
    - Defining a service URL (WS-Federation)

After configuring SSO settings, you will normally need to configure authentication credentials, the range of which depends on your SSO selection. For more information, see Configuring credentials. You might need to complete further configuration tasks for new or modified connections, depending on the selected options on the **Connection Options** tab.

Choosing SAML 2.0 profiles

A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use. SAML binding, by contrast, is the transport protocol of SAML messages.

About this task

On the **SAML Profiles** tab, select one or more SAML 2.0 profiles for your IdP Browser SSO configuration.

> ⓘ **Note**
>
> The **SAML Profiles** tab is not shown for SAML 1.x connections because identity provider (IdP) single sign-on (SSO) is assumed, single logout (SLO) profiles are not supported, and the server supports the "destination-first" (SP-initiated) profile SSO automatically. This window is also not presented for WS-Federation connections because profile selection is not required.

> ⓘ **Note**
>
> When configuring a local loopback connection, in which one PingFederate instance is both the identity provider and the service provider, disable the IdP-Initiated SLO and SP-Initiated SLO options on the Browser SSO window's SAML Profiles tab. These options determine whether SAML logout requests should be sent to the partner during the SLO flow. Those requests aren't necessary and can cause unexpected behavior when the partner connection exists locally. All local sessions for loopback connections are terminated during the SLO flow without the need to send SAML requests.

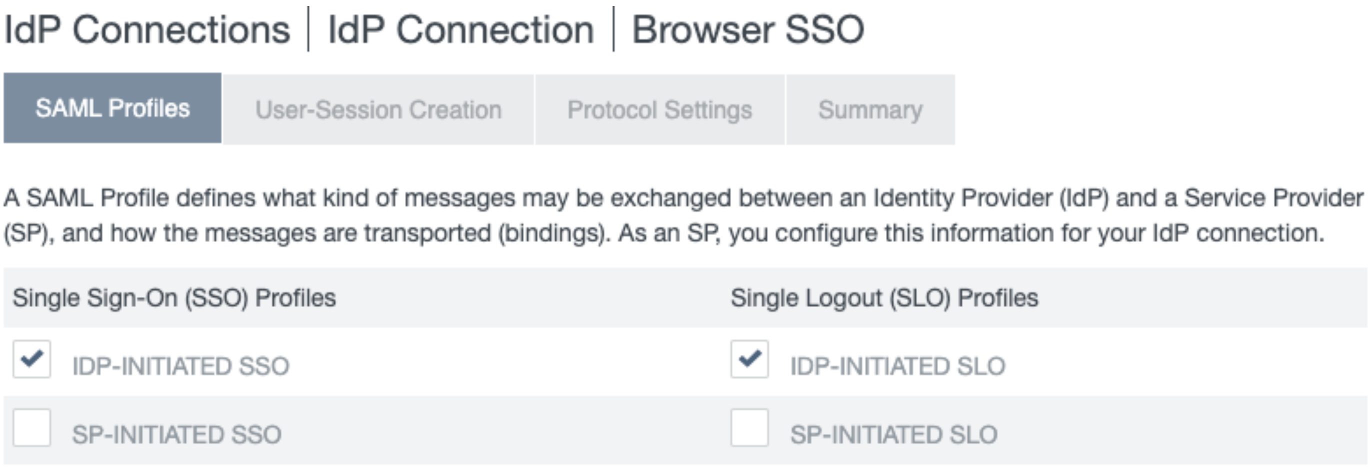
For SAML 2.0, PingFederate supports all IdP- and SP-initiated SSO and SLO profiles. For more information on typical SSO and SLO profile configurations, including illustrations, see SAML 2.0 profiles.

Steps

1. Go to **Applications** > **Integration** > **SP connections**.
2. Click on the SP connection you want to configure. For more information, see Accessing SP connections.
3. On the **Browser SSO** tab, click **Configure Browser SSO**.
4. Select either **IdP-Initiatied SSO** or **SP-Initiated SSO** or both, depending on your partner agreement.

   You must select at least one SSO profile.
5. Select either **IdP-Initiated SLO** or **SP-Initiated SLO** or both, depending on your partner agreement.

   SLO profile options are only enabled after you choose an SSO profile.



IdP Connections | IdP Connection | Browser SSO

A SAML Profile defines what kind of messages may be exchanged between an Identity Provider (IdP) and a Service Provider (SP), and how the messages are transported (bindings). As an SP, you configure this information for your IdP connection.

Single Sign-On (SSO) Profiles	Single Logout (SLO) Profiles
☑ IDP-INITIATED SSO	☑ IDP-INITIATED SLO
□ SP-INITIATED SSO	□ SP-INITIATED SLO

6. Click **Next** to save your changes.

Setting an SSO token lifetime

Identity-federation standards require a window of time during which a SSO token is considered valid. Each SSO token has an issuance time-stamp element and elements indicating the allowable lifetime of the SSO token before and after the issuance time stamp.

Before you begin

For previous steps in configuring Browser SSO, see Choosing SAML 2.0 profiles. For more information about managing service provider (SP) connections, see Accessing SP connections.

About this task

PingFederate gives you the option to change the valid lifetime of the single sign-on (SSO) token.

Steps

1. **Optional:** Override the default values for the following fields.

Field	Description
Minutes Before	The amount of time before the SSO token was issued during which it is to be considered valid.
Minutes After	The amount of time after the SSO token was issued during which it is to be considered valid.

The default value is 5 minutes for both fields.

2. Click **Next** to save your changes.

Configuring SSO token creation

As an identity provider (IdP), you must specify how PingFederate obtains user-authentication information and use it to create single sign-on (SSO) tokens appropriate for your service provider (SP) partner, including additional user attributes as needed.

About this task

If you are a federation hub bridging a service provider to one or more identity providers, you can associate one or more authentication policy contracts to the SP connection. For more information, see Federation hub use cases.

The configuration involves choosing an identity-mapping method, if applicable; establishing an attribute contract, as needed; and mapping one or more IdP adapter instances, authentication policy contracts, or both.

Steps

1. Go to **Applications** > **Integration** > **SP Connections**.
2. Click on the SP connection that you want to configure.
3. Follow the steps to reach the **Browser SSO** tab for your connection. For more information, see Configure IdP Browser SSO.
4. On the **Assertion Creation** tab, click **Configure Assertion Creation**.

Choosing an identity mapping method for IdP SSO

In the **Identity Mapping** window, you choose the type of name identifier your partner requires. Your selection might affect the way that the service provider (SP) looks up and associates your users at the SP site. You and the SP should decide in advance which option to use.

The choices of name-identifier types depend on whether you use the SAML or WS-Federation protocol. For more information, see one of the following.

- Selecting a SAML Name ID type
- Selecting a WS-Federation Name ID type

> **Note**
>
> The **Identity Mapping** window does not apply for connections using the WS-Federation protocol in conjunction with JSON web token (JWT)-based single sign-on (SSO) tokens. Instead, work with the SP to define an attribute contract that it can use to map users to accounts at the SP site.

Selecting a SAML Name ID type

You can choose a name identifier for your SAML Browser single sign-on (SSO) configuration on the **identity Mapping** tab. The type of name identifier you select affects how your service provider (SP) partner makes use of account mapping or account linking.

Before you begin

For previous steps in configuring Browser SSO, see Configure IdP Browser SSO. For more information about managing service provider (SP) connections, see Accessing SP connections.

About this task

If your SP uses account linking, establishing an attribute contract is not required. However, depending on your agreement, you can choose to supplement the account link with an attribute contract. In this configuration, the account link is used to determine the user's identity, while the additional attributes might be used for authorization decisions, customized web pages, and so on, at the SP site. For more information, see User attributes.

> **Important**
>
> If you change your configuration to use account linking without additional attributes, any existing attribute contract will be discarded in favor of the new configuration.

Steps

1. Select the type of name identifier that you and your SP have agreed to use.

Option	Description
Standard	Select if you want to send a known attribute to identify a user, for example, a username or an email address. In this scenario, the SP often uses account mapping to identify the user locally.
Pseudonym	Select if you and the SP have agreed to use a unique, opaque persistent name identifier, which cannot be traced back to the user's identity at the IdP. The SP might also use the identifier for account linking to make a persistent association between the user and a specific local account. Select the **Include attributes in addition to the pseudonym** box if you want to set up an attribute contract to use in conjunction with an opaque identifier. For more information, see Setting up an attribute contract.
Transient	Select **Transient** to enhance the privacy of a user's identity. Unlike a pseudonym, a transient identifier is different each time a user initiates SSO. An example application for this selection might be when an SP provides generalized group accounts based on organizational rather than individual identity. Select the **Include attributes in addition to the transient identifier** box if you want to set up an attribute contract to use in conjunction with an opaque identifier. For more information, see Setting up an attribute contract.

2. Click **Next** to save your changes.

Next steps

If you opted to include attributes in your name identifier, your next step will be to define the attributes. For more information, see Setting up an attribute contract. Otherwise proceed to Managing authentication source mappings.

Selecting a WS-Federation Name ID type

You can choose a name identifier for your WS-Federation Browser single sign-on (SSO) configuration on the **Identity Mapping** tab. Your selection might affect the way the service provider (SP) looks up and associates your users to their local accounts.

Before you begin

For previous steps in configuring Browser SSO, see Configure IdP Browser SSO. For more information about managing service provider (SP) connections, see Accessing SP connections.

About this task

The **Identity Mapping** window is not applicable to connections using the WS-Federation protocol in conjunction with JSON web token (JWT)-based SSO tokens. Instead, work with the SP to define an attribute contract that it can use to map users to accounts at the SP site.

Steps

1. Select the type of name identifier that you and your SP have agreed to use.

Option	Description
Email Address	This attribute is commonly used as a unique identifier for SSO and single logout (SLO). Make this selection, for example, if a user logs in using an email address or if the information is available for lookup in a local datastore.
User Principal Name	The username or other unique ID of the subject initiating the transaction. Make this selection, for example, if a username will be available from the current user session as part of a cookie or can be derived from a local datastore.
Common Name	This selection provides for anonymous SSO to your SP, generally using a hard-coded generalized sign on. Make this selection if your partner agreement involves a many-to-one use case, such as if the SP has a group account set up for all users in a particular domain.

2. Click **Next** to save your changes.

Setting up an attribute contract

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in the single sign-on (SSO) tokens for this connection.

About this task

You specify the attributes for the name identifier on your WS-Federation or, optionally, for your SAML configuration on the **Attribute Contract** tab. For more information, see Attribute contracts.

WS-Federation connections require you to define attribute contracts. For SAML connections, attribute contracts are optional if you are sending either pseudonym or transient identifiers to the partners. For more information, see Selecting a SAML Name ID type.

When establishing an attribute contract, you can change the name format when certain conditions are met. The following table summarizes the conditions and the possible actions that you can perform on the **Attribute Contract** tab.

Protocol	Identity mapping	Attribute contract	SAML_SUBJECT	Additional attributes
SAML 2.0 or SAML 1.1	Standard	Required	Built-in. Subject name format can be changed by selecting a value from a list.	Optional. Attribute name format can be changed by selecting a value from a list.
SAML 2.0 or SAML 1.1	Pseudonym or Transient	Required only if the **Include attributes ...** check box is selected on the **Identity Mapping** window. Otherwise the **Attribute Contract** window is not shown.	Assumed and cannot be added as an additional attribute.	At least one is required. Attribute name format can be changed by selecting a value from a list.
SAML 1.0	Standard	Required	Built-in. Subject name format can be changed by selecting a value from a list.	Optional. There is no attribute name format.
SAML 1.0	Pseudonym or Transient	Required only if the **Include attributes ...** check box is selected on the **Identity Mapping** window. Otherwise the **Attribute Contract** window is not shown.	Assumed and cannot be added as an additional attribute.	At least one is required. There is no attribute name format.
WS-Federation in conjunction with SAML 1.1 as the token type	Email address, user principal name, or common name	Required	Built-in. There is no subject name format.	Optional. Attribute name format can be changed by selecting a value from a list.
WS-Federation in conjunction with SAML 2.0 as the token type	Email address, user principal name, or common name	Required	Built-in. There is no subject name format.	Optional. Attribute name format can be changed by selecting a value from the list.
WS-Federation in conjunction with JWT as the token type	Not applicable	Required	Not applicable	At least one is required. There is no attribute name format.

> **Tip**
>
> If you are creating or updating a SAML service provider (SP) connection, consider using the partner’s metadata to do so. If the metadata contains the required information, PingFederate automatically populates the attribute contract for you. For more information, see Importing SP metadata.

Steps

1. Follow the required steps to create an SSO token depending on your federation protocol. For more information, see Configure IdP Browser SSO.

2. If you are using a SAML protocol, on the **Identity Mapping** tab you must select either **Pseudonym** or **Transient**, and also select the **Include Attributes** box to access the **Attribute Contract** tab. For more information, see Selecting a SAML Name ID type.

3. **Optional:** Click the **Attribute Name Format** drop-down to select a different format for the built-in subject identifier, **SAML_SUBJECT**.

   Applicable if you and the SP have agreed to a specific format. For more information, see Attribute contracts.

   > **Note**
   >
   > As needed, you can customize name-format alternatives in the `<pf_install>/pingfederate/server/default/data/config-store/custom-name-formats.xml` configuration file. Restart PingFederate to activate any changes made to this file.

4. Extend the contract with additional attributes.

   1. Enter the name of an additional attribute in the text field under **Extend the Contract**.

      Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.

      > **Tip**
      >
      > You can add a special attribute, `SAML_AUTHN_CTX`, to indicate to the SP, if required, the type of credentials used to authenticate to the identity provider (IdP) application.
      > The value of this attribute can then be mapped later on the **Attribute Contract Fulfillment** window. For more information, see Configuring contract fulfillment for IdP Browser SSO. The mapped value overrides the authentication context provided by the IdP adapter instance or the Requested AuthN Context Authentication Selector instance, through an authentication policy. If no authentication context is provided by the `SAML_AUTHN_CTX` attribute, the IdP adapter instance, or the Requested AuthN Context Authentication Selector instance, PingFederate sets the authentication context as follows:
      >
      > - For SAML 1.x `urn:oasis:names:tc:SAML:1.0:am:unspecified`
      > - For SAML 2.0 `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified`

      > **Tip**
      >
      > If you are configuring a WS-Federation connection to Microsoft Windows Azure Pack, add `upn` to the JWT’s attribute contract.

> **Tip**
>
> If you are configuring a SAML connection to an InCommon participant (see Incommon federation participants), the attribute contract might contain or require attributes such as `urn:oid:0.9.2342.19200300.100.1.3` and `urn:oid:2.5.4.42` , which are standard names under various specifications, such as RFC4524 andhttps://tools.ietf.org/html/rfc4519[RFC4519^] . The following table describes a subset of the object IDs referenced by the most common attributes used by InCommon participants.
>
> | Object ID value | Description |
> |---|---|
> | 0.9.2342.19200300.100.1.3 | mail |
> | 1.3.6.1.4.1.5923.1.1.1.6 | eduPersonPrincipalName |
> | 1.3.6.1.4.1.5923.1.1.1.7 | eduPersonEntitlement |
> | 1.3.6.1.4.1.5923.1.1.1.9 | eduPersonScopedAffiliation |
> | 1.3.6.1.4.1.5923.1.1.1.10 | eduPersonTargetedID |
> | 2.5.4.3 | cn |
> | 2.5.4.4 | sn |
> | 2.5.4.10 | o |
> | 2.5.4.42 | givenName |
> | 2.16.840.1.113730.3.1.241 | displayName |
>
> For other attributes, see the metadata from your partner. The `FriendlyName` values, if available, should provide additional information about the attributes. Alternatively, third-party resources, such as https://www.ldap.com/ldap-oid-reference and http://www.oid-info.com/, might help as well.

2. Select an attribute name format from the list.

   Applicable if you and the SP have agreed to a specific format. For more information, see Attribute contracts.

   > **Note**
   >
   > As needed, you can customize name-format alternatives in the `<pf_install>/pingfederate/server/default/data/config-store/custom-name-formats.xml` configuration file. Restart PingFederate to activate any changes made to this file.

3. Click **Add**.

4. Repeat until all desired attributes are defined.

5. **Optional:** Click **Edit** to change the configuration of an existing attribute.

6. **Optional:** Click **Delete** to remove an existing attribute.

7. Click **Next** to save changes.

Managing authentication source mappings

On the **Authentication Source Mapping** tab, you can map identity provider (IdP) adapters and authentication policies to authenticate users to your service provider (SP).

About this task

IdP adapters are responsible for handling user authentication as part of an single sign-on (SSO) operation. A configured adapter in PingFederate is known as an adapter instance.

In a basic scenario, you map an IdP adapter instance to a SP connection on the **Authentication Source Mapping** tab and complete its mapping configuration through a series of sub tasks. When a user starts an SSO request, the corresponding IdP adapter is triggered to authenticate the user. Upon successful authentication, PingFederate creates and sends an SSO token to the SP based on the connection settings. As needed, you can map multiple IdP adapter instances to an SP connection, the same IdP adapter instance to multiple SP connections, or a combination of them.

If you use authentication policies to route users through a series of authentication sources and end each successful policy path with an authentication policy contract (APC), you can map the APC to your connection. Like IdP adapter instances, you can map multiple APCs to an SP connection, the same APC to multiple SP connections, or a combination of them.

> **Tip**
>
> For more information about authentication policies and contracts, see Authentication policies.

You can also map one or more APCs to an SP connection to bridge a service provider to one or more identity providers. In this scenario, PingFederate is a federation hub for both sides. PingFederate uses APCs to associate this SP connection with the applicable IdP connections to the identity providers. Each APC has its own set of attributes which you map values to the SSO tokens.

> **Tip**
>
> For more information about the federation hub, see Federation hub use cases.

Regardless of how many IdP adapter instances and APCs are mapped to an SP connection, PingFederate uses only one adapter instance or policy path to authenticate a user. You can leave the decision to the users or create authentication policies to mandate authentication requirements. Because each adapter instance or APC could return different user attributes, each mapping must define how the attribute contract is fulfilled in its mapping configuration.

Steps

1. For initial steps to configure SP connections, see Accessing SP connections.
2. For initial steps to configure Browser SSO, see Configure IdP Browser SSO.
3. For initial steps to configure assertion creation. see Configuring SSO token creation.
4. On the **Authentication Source Mapping** tab, select one of the following.

Choose from:

- Click **Map New Adapter Instance** to map a new IdP Adapter instance. For more information, see Mapping an adapter instance.
- Click **Map New Authentication Policy** to map a new APC. For more information, see Mapping an authentication policy
- Click on an existing instance to edit its configuration.
- Click **Delete** to remove an existing adapter instance or APC. Click **undelete** to cancel the removal request

When authentication sources, such as IdP adapter instances or connection mapping contracts, are restricted to certain virtual server IDs, the allowed IDs are displayed under **Virtual Server IDs**.

5. When your authentication sources have been mapped, click **Next** save your changes.

Mapping an adapter instance

After extending your attribute contract, you can map adapter instances on the **Authentication Source Mapping** tab.

Steps

1. For initial steps, see Managing authentication source mappings.
2. On the **Authentication Source Mapping** tab, click **Map New Adapter Instance**.
3. On the **Adapter Instance** tab, select an adapter instance from the **Adapter Instance** drop-down list.
4. **Optional:** If you want to customize one or more adapter settings for this connection alone, select the **Override Instance Settings** check box.

> **Note**
>
> If you are editing a currently mapped adapter instance, you can toggle **Override Instance Settings**. Clearing it removes all previously overridden settings for this connection. Selecting it provides you the opportunity to customize adapter settings specifically for this connection.

> **Tip**
>
> Alternatively, you can create child adapter instances of a base adapter instance (with overrides) so that customized settings can be applied to several connections. For more information, see Hierarchical plugin configurations.

Result:

Selecting the **Override instance settings** box will add the **Override Instance** tab to the navigation bar. For more information, see Overriding an IdP adapter instance

5. Click **Next**, and refer to the following topics to complete the configuration.

Related links

- Restricting an authentication source to certain virtual server IDs
- Selecting an attribute mapping method

- Configuring default contract fulfillment for IdP Browser SSO
- Defining issuance criteria for IdP Browser SSO
- Reviewing the authentication source mapping

Mapping an authentication policy

After extending your attribute contract, the **Authentication Source Mapping** tab gives you the option to map an authentication policy.

Steps

1. Click **Map New Authentication Policy**.
2. On the **Authentication Policy Contract** tab, select a contract from the **Authentication policy contract** list.

   If you need to create a new contract or manage an existing contract, click **Manage Policy Contracts**.
3. Click **Next**, and refer to the following topics to complete the configuration.

Related links

- Restricting an authentication source to certain virtual server IDs
- Selecting an attribute mapping method
- Configuring default contract fulfillment for IdP Browser SSO
- Defining issuance criteria for IdP Browser SSO
- Reviewing the authentication source mapping

Overriding an IdP adapter instance

On the **Override Instance** window, you can start a series of sub tasks to override adapter settings specifically for this connection.

About this task

> **Note**
>
> Any changes to the base adapter instance are propagated to a connection as long as you don't override those changes.

Steps

1. For initial steps to configure authentication source mapping, see Managing authentication source mappings.
2. On the **Adapter Instance** tab, click **Override Instance Settings**.
3. On each of the settings tabs, select the **Override** check box, make your changes, and then click **Next**.

   > **Note**
   >
   > If you are editing a currently mapped adapter instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You can also remove all overridden settings on a per-window basis by clearing the **Override** check box near the top of the window.

The override setting windows are functionally identical to those used for creating a new adapter instance. For more information, see Managing IdP adapters.

4. When you are finished, click **Done** to proceed to Selecting an attribute mapping method.

Restricting an authentication source to certain virtual server IDs

On the **Virtual Server IDs** tab, when you multiplex one connection for multiple environments, you can enforce authentication requirements by restricting an authentication source to certain virtual server IDs.

About this task

Authentication sources are unrestricted by default. For more information, see Multiple virtual server IDs

Steps

1. Select the **Restrict Virtual Server IDs** check box.

2. Select one or more virtual server IDs that you want to allow for this authentication source.

Result

If you are editing a currently mapped adapter instance or authentication policy contract (APC), you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Selecting an attribute mapping method

On the **Mapping Method** tab, you can select if and how PingFederate should query local datastores to help fulfill the attribute contract in conjunction with attribute values from the authentication source.

About this task

To determine whether you need to look up additional values, compare the attribute contract against the adapter contract or the authentication policy contract. If the attribute contract requires more information, you must determine whether local datastores can supply it.

> **Tip**
>
> Alternatively, you can configure datastore queries as part of the fulfillment configuration for the applicable identity provider (IdP) adapter contract or authentication policy contract. If so, you do not need to set up datastore query on the connection level.
> For more information, see Defining the IdP adapter contract or Applying policy contracts or identity profiles to authentication policies.

Steps

1. For initial steps to configure IdP adapter instances, see Mapping an adapter instance.

2. On the **Mapping Method** tab, select one of the following options.

Mapping method	Description
Retrieve additional attributes from multiple data stores using one mapping	Select to configure one or more datastores to look up attributes for a single mapping.

Mapping method	Description
Retrieve additional attributes from a data store	Select to define alternate datastores to look up attributes and a failsafe mapping configuration. ⓘ **Note** When this option is selected, the token authorization framework, through issuance criteria, does not apply. For more information, see Token authorization and Selecting an attribute mapping method.
Use only the adapter contract values in the SAML assertion	Select if you do not require connection-level datastore query.

3. Click **Next** to save changes and proceed to the next tab.

   If you opted to require datastore queries, see Configuring attribute sources and user lookup. If not, see Configuring contract fulfillment for IdP Browser SSO.

Configuring default contract fulfillment for IdP Browser SSO

On the **Attribute Contract Fulfillment** tab, you can define the default attributes PingFederate will send to the service provider (SP) in case of failure to complete the attribute contract.

Before you begin

For initial steps to configure identity provider (IdP) adapter instances or authentication policy contracts (APC), see Managing authentication source mappings.

If you have selected the failsafe option on the **Mapping Method** tab and the **Send user to SP using default list of attributes** option on the **Failsafe Attribute Source** tab, define the default values that should be sent in the single sign-on (SSO) tokens to the SP.

About this task

On the **Attribute Contract Fulfillment** tab, you must complete the following steps for each adapter instance or APC.

Steps

1. Select a source from the **Source** drop-down list.
2. Select a source from the **Source** list and then choose or enter a value. You must map all attributes.
   - **Adapter** or **Authentication Policy Contract** (the authentication source)

     When selected, the **Value** list is populated with attributes from the authentication source. Select the desired attribute from the list. At runtime, the attribute value from the authentication source is mapped to the value of the attribute in the SSO token.

     For example, to map the value of the HTML Form Adapter's `username` attribute as the value of the `SAML_SUBJECT` attribute on the contract, select **Adapter** from the **Source** list and **username** from the **Value** list.
   - **Context**

When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the SSO token.

> **Important**
>
> If you are configuring an SP connection to bridge one or more identity providers to a service provider, consider mapping the original issuer of the assertions into an attribute by selecting **Context** as the source and **Authenticating Authority** as the value. This is important when bridging multiple identity providers to one service provider, where the service provider should take the information about the original issuer into consideration before granting access to protected resources.
> For more information, see Bridging multiple IdPs to an SP.

> **Note**
>
> Because the **HTTP Request** context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values (see **Expression**).

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

  Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  Select this option to ignore the **Value** field, causing no value selection to be necessary.

- **Text**

  When selected, the text you enter is mapped to the value of the attribute in the single sign-on tokens at runtime. You can mix text with references to any of the values from the authentication source using the `$\{attribute}` syntax.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}` . The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

> **Tip**
>
> Two other text variables are also available: `${SAML_SUBJECT}` and `${TargetResource}` . `SAML_SUBJECT` is the initiating user (or other entity). `TargetResource` is a reference to the protected application or other resource for which the user requested SSO access; the `${TargetResource}` text variable is available only if specified as a query parameter for the relevant endpoint (either as `TargetResource` for SAML 2.0 or `TARGET` for SAML 1.x).

3. After all attributes have been mapped, click **Next** to save changes.

Defining issuance criteria for IdP Browser SSO

Configure the criteria that PingFederate uses to determine user authorization to access service provider (SP) resources.

About this task

On the **Issuance Criteria** tab, define the criteria that must be satisfied in order for PingFederate to process a request further. This token authorization feature provides the capability to conditionally approve or reject requests based on individual attributes.

> **Note**
>
> The **Issuance Criteria** tab does not appear if you have chosen the failsafe option on the **Mapping Method** tab. For more information, see Selecting an attribute mapping method.

Begin this optional configuration by adding a criterion. Choose the source that contains the attribute to be verified. Some sources, such as **Mapped Attributes**, are common to almost all use cases. Other sources, such as **JDBC**, depend on the type of configuration. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to be verified. Depending on the selected source, the available attributes or properties vary. Finally, specify the comparison method and the desired, compared-to, value.

If you define multiple criteria, all criteria must be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value depending on the chosen comparison method. The multi-value contains and multi-value does not contain comparison methods are intended for attributes that might contain multiple values. Such criterion is considered satisfied if one of the multiple values matches or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Important**
>
> When you multiplex one connection for multiple environments, consider using attribute mapping expressions to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access. For more information, see Multiple virtual server IDs and Issuance criteria and multiple virtual server IDs.

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

Source	Description
Adapter or **Authentication Policy Contract**	Select to evaluate attributes from an identity provider (IdP) adapter instance or an authentication policy contract.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

1. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

1. In the **Value** field, enter the comparison value.

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

Error results are handled differently for IdP-initiated single sign-on (SSO) and SP-initiated SSO requests.

IdP-initiated SSO

Redirect

When an `InErrorResource` URL is provided, the value of the **Error Result** field is used by an `ErrorDetail` query parameter in the redirect URL.

Template

When an `InErrorResource` URL is not provided, the value of the **Error Result** field is used by the variable *$errorDetail* in the `idp.sso.error.page.template.html` template file.

SP-Intiated SSO

SAML

The **Error Result** field value is used by the `StatusMessage` element in the response to the SP.

WS-Federation (Template)

The **Error Result** field value is used by the *$errorDetail* variable in the `<pf_install>/pingfederate/server/default/conf/template/sourceid-wsfed-idp-exception-template.html` template file.

Using an error code in the **Error Result** field allows the error template or an application to process the code in a variety of ways. For example, the template or application can display an error message or e-mail an administrator.

To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

2. Click **Add**.

3. **Optional:** Repeat to add more criteria.

4. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.

   2. In the **Expression** field, enter the required expressions.

   3. In the **Error Result** field, enter an error code or message.

> **Note**
>
> If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click **Add**.

5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.

6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Configuring attribute sources and user lookup

Attribute sources are specific datastore or directory locations containing information that might be needed for the attribute contract. You can use more than one attribute source when mapping values to the attribute contract.

About this task

The order in which attribute sources are listed affects the queries differently based on the selection made on the **Mapping Method** tab. For more information, see Selecting an attribute mapping method.

Retrieve additional attributes from multiple data stores using one mapping

If you plan on using the result of a query as an input to a subsequent query, stack your attribute sources accordingly.

Retrieve additional attributes from a data store

As soon as a query succeeds, PingFederate moves on to the next task, contract fulfillment. Therefore you should prioritize the attribute sources.

Steps

1. Click **Add Attribute Source** and then follow a series of sub tasks to complete the configuration.

2. See Choosing a datastore for instructions on configuring and adding attribute sources.

3. Repeat as necessary to add additional sources.

Result

If you are editing a currently mapped adapter instance or authentication policy contract, you can add, remove, or reorder attribute sources, which might require additional configuration changes in subsequent tasks.

Configuring contract fulfillment for IdP Browser SSO

On the **Attribute Contract Fulfillment** tab, you can map values to the attributes defined for the contract. These are the values that will be included in the single sign-on (SSO) tokens sent to the service provider (SP).

Before you begin

For initial steps to configure identity provider (IdP) adapter instances or authentication policy contracts (APC), see Managing authentication source mappings.

About this task

If you are bridging one or more identity providers to a service provider, map values to an authentication policy contract. For more information, see Federation hub use cases.

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

On the **Attribute Contract Fulfillment** tab, you must complete the following steps for each attribute contract.

Steps

1. Select a **Source** from the drop-down.
2. Select a **Value** from the drop-down or enter a **Value** in the text field. See the following for more information.
   - **Adapter** or **Authentication Policy Contract** (the authentication source)

     When selected, the **Value** list is populated with attributes from the authentication source. Select the desired attribute from the list. At runtime, the attribute value from the authentication source is mapped to the value of the attribute in the SSO token.

     For example, to map the value of the HTML Form Adapter's `username` attribute as the value of the `SAML_SUBJECT` attribute on the contract, select **Adapter** from the **Source** list and **username** from the **Value** list.
   - **Context**

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the SSO token.

     > **Important**
     >
     > If you are configuring an SP connection to bridge one or more identity providers to a service provider, consider mapping the original issuer of the assertions into an attribute by selecting **Context** as the source and **Authenticating Authority** as the value. This is important when bridging multiple identity providers to one service provider, where the service provider should take the information about the original issuer into consideration before granting access to protected resources.
     > For more information, see Bridging multiple IdPs to an SP.

     > **Note**
     >
     > Because the **HTTP Request** context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values (see **Expression**).
   - **LDAP**, **JDBC**, or **Other**

     When selected, the **Value** list populates with attributes that you have selected in the attribute source configuration. Select the desired attribute from the list. At runtime, the attribute value from the attribute source is mapped to the value of the attribute in the SSO token.
   - **Expression**

     When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  Select this option to ignore the **Value** field.

- **Text**

  When selected, the text you enter is mapped to the value of the attribute in the single sign-on tokens at runtime. You can mix text with references to any of the values from the authentication source using the `$\{attribute}` syntax.

  You can also enter values from your datastore, when applicable, using this syntax:

  `[.codeph]``$\{ds.[.varname]__attr-source-id.attribute__}```

  where `attr-source-id` is the attribute source ID value and `attribute` is one of the selected attributes in the attribute source configuration.

  Note that when using alternate data stores with a failsafe mapping, the attribute source ID value is not applicable. Use the following syntax instead:

  `[.codeph]``$\{ds.[.varname]__attribute__}```

  > **Tip**
  >
  > Two other text variables are also available: `${SAML_SUBJECT}` and `${TargetResource}`. `SAML_SUBJECT` is the initiating user (or other entity). `TargetResource` is a reference to the protected application or other resource for which the user requested SSO access; the `${TargetResource}` text variable is available only if specified as a query parameter for the relevant endpoint (either as `TargetResource` for SAML 2.0 or `TARGET` for SAML 1.x).

There are also a variety of reasons why you might hard code a text value. For example, if the SP web application provides a service based on the name of your organization, you might provide that attribute value as a constant.

+

3. After all attributes have been mapped, click **Next** to save changes.

   > **Note**
   >
   > If you are editing a currently mapped adapter instance or APC, you can update the mapping configuration, which might require additional configuration changes in subsequent tasks.

Reviewing the authentication source mapping

You can review and save your authentication source mapping in the **Summary** tab.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Reviewing the SSO token creation summary

Review and save your single sign-on (SSO) token creation configuration in the **Summary** tab.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Configuring protocol settings

The **Protocol Settings** tab provides the launching point for configuring partner endpoints, message customizations, and other protocol-specific settings for browser-based single sign-on (SSO) connections.

Before you begin

For initial steps to configure a service provider (SP) connection, see Accessing SP connections.

For initial steps to configure Browser SSO, see Configure IdP Browser SSO.

About this task

SAML 2.0

- Outbound SSO bindings (POST, artifact) and the corresponding assertion consumer service (ACS) URLs
- Outbound SLO bindings (POST, redirect, artifact, SOAP) and the corresponding protocol endpoints
- Inbound bindings (POST, redirect, artifact, SOAP)
- Artifact lifetime
- Signature policy

- Encryption policy

SAML 1.x

- Outbound SSO bindings (POST, artifact) and the corresponding assertion consumer service (ACS) URLs
- Default target URL
- Artifact lifetime
- Signature policy

WS-Federation

- Protocol endpoint
- Default target URL

Steps

1. Before configuring Browser SSO protocol settings, you must first configure assertion configuration. For more information, see Configuring SSO token creation
2. In the **Protocol Settings** tab, click **Configure Protocol Settings** to begin.

Setting Assertion Consumer Service URLs (SAML)

If your PingFederate configuration uses any version of SAML, you can configure assertion indexes, bindings, and endpoint URLs on the **Assertion Consumer Service URL** tab.

Before you begin

For prerequisites and initial steps to configure Browser SSO protocol settings, see Configuring protocol settings.

About this task

The assertion consumer service (ACS) endpoint is a location to which the single sign-on (SSO) tokens are sent, according to partner requirements. ACS is applicable to all SAML versions and both the identity provider (IdP)- and service provider (SP)-initiated SSO profiles.

> **Note**
>
> The SP might request that the SAML assertion be sent to one of several URLs, using different bindings. PingFederate uses the defined URL entries on this page to validate the authentication request. However, per SAML specifications, if the request is signed, PingFederate can verify the signature instead. The ACS URL does not necessarily need to be listed here. This is useful for scenarios where an ACS URL might be dynamically generated.

Some federation use cases might require additional customizations in the assertions sent from the PingFederate IdP server to the SP, such as placing well-formed XML in the `<AttributeValue>` element or including the optional `SessionNotOnOrAfter` attribute in the `<AuthnStatement>` element. You can use OGNL expressions to fulfill these use cases.

Steps

1. In the **Assertion Consumer Service URL** tab, configure one or more SAML ACS endpoints.

    1. Select a SAML binding from the **Binding**drop-down list.

    2. Enter the ACS endpoint URL to the **Endpoint URL** field.

        You can enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** window.

    3. **Optional:** Select the **Default** box if you want this entry to be the default ACS endpoint.

        The administrative console always sets the first entry as the default ACS endpoint. You can reset the default endpoint when you add ACS endpoint.

    4. **Optional:** Enter an integer to the **Index** field for this ACS endpoint.

        The administrative console automatically assigns an index value for each ACS endpoint, starting from 0. If you want to define your own index values, you must make sure the index values are unique.

    5. Click **Add**.

    6. **Optional:** Repeat to add additional ACS endpoints.

2. **Optional:** Customize messages using OGNL expressions.

    > **Note**
    >
    > OGNL expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

    1. Click **Show Advanced Customizations**.

    2. Select a message type from the list.

    3. Enter an OGNL expression to fulfill your use case.

        > **Note**
        >
        > For more information about **Message Type**, available variables, and sample OGNL expressions, see Customizing assertions and authentication requests.

    4. Click **Add**.

    5. **Optional:** Repeat to add another message customization.

3. Click **Next** to proceed to the next tab. For SAML 1.x configurations, see Setting a default target URL (SAML 1.x). For SAML 2.0, see Specifying SLO service URLs (SAML 2.0).

Result

If you are editing an existing connection, you can reconfigure any items, which could require additional configuration changes in subsequent tasks. You must always configure at least one ACS endpoint.

Setting a default target URL (SAML 1.x)

SAML 1.x service provider (SP) connections require that a default target URL be specified for a scenario where the identity provide (IdP) application does not include one in its single sign-on (SSO) request. This default URL represents the destination on the SP where the user will be directed.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

Steps

1. Enter default destination in the **Default Target URL** field.

2. Click **Next** to save your setting.

Result

If you are editing an existing connection, you update the target destination. You must always define a default destination when configuring a SAML 1.x SP connection.

Specifying the WS-Trust version

You can specify whether to use WS-Trust version 1.2 or 1.3 for tokens. The default version is 1.2.

Before you begin

Learn more about prerequisites and initial steps for configuring Browser SSO protocols in Configuring protocol settings.

Steps

1. In the **WS-Trust Version** list, select version **1.2** or **1.3**.

> **Note**
>
> For version 1.3, the response is always a `RequestSecurityTokenResponseCollection` object as in the following example.
>
> ```
> <wst:RequestSecurityTokenResonseCollection xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">
> ```
>
> Learn more about WS-Trust version 1.3 in OASIS WS-Trust 1.3 Standards.

2. Click **Next** to save your changes.

Defining a service URL (WS-Federation)

On the **Service URL** tab, you can enter the WS-Federation protocol endpoint of your service provider (SP) partner where PingFederate will send single sign-on (SSO) tokens and single logout (SLO) cleanup messages.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

The SSO tokens are transmitted within a Request for Security Token Response (RSTR) message in response to a request for authentication from the SP. SLO cleanup messages are sent to your partner when PingFederate, as the identity provider (IdP), receives a user's SLO request. These cleanup messages indicate that the user's local session has been terminated.

To protect against session token hijacking, you can specify additional allowed domains and paths on this window. If the option to validate `wreply` for SLO is enabled, these additional domains and paths will also be taken into consideration as well. For more information, see Managing partner redirect validation.

Some federation use cases might require additional customizations in the RSTR message sent from the PingFederate IdP server to the SP. You can use OGNL expressions to fulfill these use cases.

Steps

1. On the **Service URL** tab, enter the WS-Federation protocol endpoint at the SP site in the **Endpoint URL** field.

   You can enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** tab. For more information, see Identifying the SP.

2. **Optional:** Specify additional allowed domains and paths.

   Indicate whether to mandate secure connections when this resource is requested under **Require HTTPS**.

   > **Important**
   >
   > This selection is recommended to ensure that the validation will always prevent message interception for this type of potential attack, under all conceivable permutations.

   This check box is selected by default.

   Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

   Enter a value without the protocol, such as `example.com` or `10.10.10.10`.

   Prefix a domain name with a wildcard followed by a period to include subdomains using one entry. For instance, `*.example.com` covers `hr.example.com` or `email.example.com` but not `example.com`, the parent domain.

   > **Important**
   >
   > While using an initial wildcard provides the convenience of allowing multiple subdomains using one entry, consider adding individual subdomains to limit the redirection to a list of known hosts.

   Enter the exact path of this resource under **Valid Path**.

Start with a forward slash, without any wildcard characters in the path. If left blank, any path under the specified domain or IP address is allowed. This value is case-sensitive. For instance, `/inbound/Consumer.jsp` allows `/inbound/Consumer.jsp` but rejects `/inbound/consumer.jsp`.

You can allow specific query parameters with or without a fragment by appending them to the path. For instance, `/inbound/Consumer.jsp?area=West&team=IT#ref1001` matches `/inbound/Consumer.jsp?area=West&team=IT#ref1001` but not `/inbound/Consumer.jsp?area=East&team=IT#ref1001`.

Select the check box under **Allow Any Query/Fragment** to allow any query parameters or fragment for this resource.

Selecting this check box also means that no query parameter and fragment are allowed in the path defined under **Valid Path**.

This check box is not selected by default.

Click **Add**.

Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Use the **Delete** and **Undelete** workflow to remove an existing entry or cancel the removal request.

Repeat these steps to define multiple expected resources. NOTE: The display order does not matter. A more specific match is considered a better match and an exact match is considered the best match.

3. **Optional:** Customize messages using OGNL expressions.

   Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

   1. Click **Show Advanced Customizations**.
   2. Select a message type from the list.
   3. Enter an OGNL expression to fulfill your use case.

      > **Note**
      >
      > For more information about **Message Type**, available variables, and sample OGNL expressions, see Customizing assertions and authentication requests.

   4. Click **Add**.
   5. **Optional:** Repeat to add another message customization.
4. Click **Next** to save your changes.

Result

If you are editing an existing connection, you can reconfigure any items, which might require additional configuration changes in subsequent tasks.

Specifying SLO service URLs (SAML 2.0)

On the **SLO Service URLs** tab, you associate bindings to the endpoints where your service provider (SP) receives logout requests when single logout (SLO) is initiated at your site and where PingFederate sends SLO responses when it receives SLO requests from the SP.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

This step applies only to SAML 2.0 connections when either SLO profile is selected on the **SAML Profiles** tab. For more information, see Choosing SAML 2.0 profiles.

Steps

1. Select a SAML binding from the list.
2. Enter the SLO endpoint URL to the **Endpoint URL** field.

   You can enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** tab. For more information, see Identifying the SP.
3. **Optional:** Enter a URL in the **Response URL** field.

   When specified, this URL is the location to which SLO logout response messages are sent based on your partner agreement. When omitted, PingFederate sends logout responses to the SLO endpoint URL.

   You can enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** window. For more information, see Identifying the SP.
4. Click **Add**.
5. **Optional:** Repeat to add additional SLO endpoints.
6. Click **Next** to save your settings.

Result

If you are editing an existing connection, you can reconfigure the SLO endpoints, which might require additional configuration changes in subsequent tasks.

Choosing allowable SAML bindings (SAML 2.0)

On the **Allowable SAML Bindings** tab, you select the one or more bindings that your service provider (SP) partner can use to send SAML authentication requests or single logout (SLO) messages.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

This step applies only to SAML 2.0 connections when the SP-initiated SSO profile or either SLO profile is selected on the **SAML Profiles** tab.

Steps

1. On the **Allowable SAML Bindings** tab, select the applicable SAML bindings based on your partner agreement.

   > ⓘ **Note**
   >
   > If you have specified an Assertion Consumer Service (ACS) or SLO endpoint using the artifact (outbound) binding, you must including SOAP as one of the allowable (inbound) binding.

2. Click **Next** to save changes and proceed to **Artifact Resolver Locations**. For more information, see Specifying artifact resolver locations (SAML 2.0).

Result

If you are editing an existing connection, you can reconfigure the allowable bindings, which might require additional configuration changes in subsequent tasks.

Setting an artifact lifetime (SAML)

When PingFederate sends an artifact to your service provider (SP)'s SAML ACS endpoint or SAML 2.0 SLO endpoint, an element in the message indicates how long it should be considered valid. On the **Artifact Lifetime** tab, you can specify the expiry information in seconds.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

You can change the default value to meet your requirements. You should also consider synchronizing your serve clock with your partner’s SAML gateway server. If clocks are not synchronized, you might need to set the artifact lifetime to a higher value to prevent latency issues.

Steps

1. **Optional:** On the **Artifact Lifetime** tab, override the default value of the **Artifact Lifetime** field.

   The default value is 60 (seconds).

2. Click **Next** to save your changes.

Specifying artifact resolver locations (SAML 2.0)

When the artifact binding is enabled as one of the allowable bindings on the **Allowable SAML Bindings** tab, you must provide at least one artifact resolution service (ARS) endpoint on the **Artifact Resolver Locations** tab.

About this task

The ARD endpoint is where PingFederate sends back-channel requests to resolve artifacts received from the service provider (SP).

Steps

1. On the **Artifact Resolver Locations** tab, enter the ARS endpoint URL in the **URL** field.

   You can enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** tab. For more information, see Identifying the SP.

2. **Optional:** Enter an integer to the **Index** field for this ACS endpoint.

   The administrative console automatically assigns an index value for each ARS endpoint, beginning with 0. If you want to define your own index values, you must make sure the index values are unique.

3. Click **Add**.

4. **Optional:** Repeat to add additional ARS endpoints.

   > **Note**
   >
   > When specifying multiple ARS endpoints, each endpoint must share the same transport protocol. That is, if one endpoint uses HTTPS, then all must use HTTPS.

5. After you have entered all of your ARS endpoints, click **Next** to save changes.

Result

If you are editing an existing connection, you can reconfigure any ARS endpoints.

Defining signature policy (SAML)

On the **Signature Policy** tab, you can control how digital signatures are used for SAML messages.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

The choices made in this tab depend on your partner agreement and your federation protocol. For more information, see Digital signing policy coordination.

SAML 2.0

Digital signing is required for SAML response messages sent from the identity provider (IdP) with the POST or redirect binding. Based on the SAML specifications, PingFederate provides three options:

- Select **Always Sign Assertion** to always sign the assertion portion inside the SAML response message.
- Select **Sign Response As Required** to sign the SAML response message per the SAML specifications. This is the default selection.
- Select both to always sign the assertion portion inside the SAML response message for all bindings and to sign the SAML response message per the SAML specifications.

Authentication request messages from the service provider (SP) may also be signed to enforce security. This scenario applies only when the SP-initiated single sign-on (SSO) profile is enabled on the **SAML Profiles** tab. Select**Require Authn Requests to be Signed** to enforce this digital signature requirement. For more information, see Choosing SAML 2.0 profiles.

SAML 1.x

For SAML 1.0 and SAML 1.1, the assertion portion inside the SAML response message can be digitally signed.

- Select **Always Sign Assertion** to always sign the assertion portion inside the SAML response message.

Steps

1. On the **Signature Policy** tab, select the options based on your partner agreement and federation protocol.
2. Click **Next** to save changes.

Result

If you are editing an existing connection, you can reconfigure the digital signature policy, which might require additional configuration changes in subsequent tasks.

Configuring XML encryption policy (SAML 2.0)

For SAML 2.0 configurations, in addition to using signed assertions to ensure authenticity, you and your partner can also agree to encrypt all or part of an assertion to improve privacy. If so, you can configure these settings on the **Encryption Policy** tab.

Before you begin

For prerequisites and initial steps for configuring Browser SSO protocols, see Configuring protocol settings.

About this task

> **Note**
>
> For WS-Fed connections with SAML 2.0 assertions, you cannot encrypt the entire assertion.

Option	Name identifier (SAML_SUBJECT)	Other attributes	Encrypt the SAML_SUBJECT in SLO messages to the SP	Allow encryption in SLO messages from the SP
None	No encryption.	No encryption.	No encryption.	No encryption.
The entire assertion	Encrypted.	Encrypted.	Available as an option.	Available as an option.
One or more attributes	Available as an option.	Available as an option.	Available as an option only if you select to encrypt the name identifier (SAML_SUBJECT).	Available as an option only if you select to encrypt the name identifier (SAML_SUBJECT).

Steps

1. Select the options based on your partner agreement.
2. Click **Next** to save changes.

Result

If you are editing an existing connection, you can reconfigure the XML encryption policy, which might require additional configuration changes in subsequent tasks.

Reviewing protocol settings

On the **Summary** tab, you can review and save your protocol settings.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Reviewing browser-based SSO settings

On the **Summary** tab, you can review and save your browser-based single sign-on (SSO) settings.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Configuring the Attribute Query profile in an SP connection

At the Attribute Query step, you configure your connection to respond to requests for user attributes from your partner service provider (SP), if you have chosen this option.

Before you begin

For prerequisites and previous steps to configure identity provider (IdP) Browser single sign-on (SSO), see Configure IdP Browser SSO.

About this task

For more information about the Attribute Query profile, see Choosing SP connection options.

Attribute queries do not depend on SSO, but can be used independently or in conjunction with Browser SSO or provisioning to provide flexibility in how a user authenticates with SP applications. For more information, see Attribute Query and XASP.

Steps

1. On the **Attribute Query** tab, click **Configure Attribute Query Profile**. See Defining retrievable attributes for next steps.

Defining retrievable attributes

On the **Retrievable Attributes** window, you specify the user attributes you and your partner have agreed to allow in an attribute query transaction.

Before you begin

For prerequisites and previous steps to configure the Attribute Query profile, see Configuring the Attribute Query profile in an SP connection.

About this task

> **Note**
>
> The service provider (SP) might not necessarily request all of these attributes in each attribute-query request. Instead, the list simply limits the request to a subset of these attributes.

Steps

1. On the **Retrievable Attributes** tab, follow these steps to configure your Attribute Query attributes. To add an attribute, enter the attribute name in the text box and then click **Add**.

   > **Note**
   >
   > Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.

2. To modify an attribute name:

Action	Steps
Add an attribute	Enter the attribute name in the text box. Click **Add**.
Modify an existing attribute	Click **Edit**. Modify the attribute name in the text box. Click **Update**.
Delete an existing attribute	Click **Delete**.

3. Click **Next** to save your changes.

Configuring attribute lookup

The optional attribute lookup configuration allows you to configure one or more datastores to look up attributes and to set up search parameters.

Before you begin

For prerequisites and previous steps to configure the Attribute Query profile, see Configuring the Attribute Query profile in an SP connection.

About this task

Attribute sources are specific datastore or directory locations containing information that is returned to the service provider (SP) in response to an attribute request.

Steps

1. On the **Attribute Sources & User Lookup** tab, you can do the following

   Choose from:

   - To configure an attribute source, click **Add Attribute Source** and complete the setup steps. For more information, see Choosing a datastore for Attribute Query.
   - To modify an attribute source configuration, select the attribute source and complete the setup steps.

     > **Note**
     >
     > Depending on what you change, you might need to modify dependent data in subsequent steps, as indicated.

2. When your attribute sources are configured, click **Next** to save your changes.

Choosing a datastore for Attribute Query

On the **Data Store** tab, choose a datastore instance for PingFederate to look up attributes.

Before you begin

For prerequisites and previous steps to configure the Attribute Query profile, see Configuring the Attribute Query profile in an SP connection.

About this task

The process of configuring PingFederate to look up attributes in a datastore for attribute-query responses is similar to that used for single sign-on (SSO) Attribute Sources and User Lookup.

Steps

1. Enter a **Description** for the datastore in the text box.
   1. If prompted, enter an**ID** in the text box.
2. Select a datastore instance from the **Active Data Store** list.

> **Tip**
>
> If the datastore you want is not shown in the **Active Data Store** list, click **Manage Data Stores** to review or add a datastore instance. For more information, see Datastores.

3. Depending on the datastore type, the rest of the setup varies as follows.

Data store type	Required tasks
JDBC	◦ Specifying database tables and columns ◦ Entering a database search filter
LDAP	◦ Specifying directory properties and attributes ◦ Defining encoding for binary attributes (optional) ◦ Entering a directory search filter
Other	◦ Specifying data source filters and fields

> **Important**
>
> When attribute queries are sent using X.509 Attribute Sharing Profile (XASP), use the variable `${SubjectDN}` —rather than `${SAML_SUBJECT}` —to retrieve the subject identifier.
> You can also use any of these distinguished name (DN)-parsing variables:
>
> - `${CN}`
> - `${OU}`
> - `${O}`
> - `${L}`
> - `${S}`
> - `${C}`
> - `${DC}`
>
> If more than one value exists for any of the parsing variables, then they are enumerated. For example, if the Subject DN is `cn=John Smith,ou=service,ou=employee`, then you could use any of these elements in your filter qualifier:
>
> - `${SubjectDN}=cn=John Smith,ou=service,ou=employee`
> - `${ou}=service`
> - `${ou1}=employee`
>
> For more information about XASP, see Attribute Query and XASP.

4. When you have finished configuring your datastore, click **Next** to save changes.

Configuring mapping fulfillment for Attribute Query

The last step in configuring an attribute source is to map values into the assertion to be sent in response to an attribute query on the **Attribute Mapping Fulfillment** tab.

Before you begin

For prerequisites and previous steps to configure the Attribute Query profile, see Configuring the Attribute Query profile in an SP connection.

Steps

1. For each attribute, select a source from the **Source** list and then choose or enter a value.

   - **Context**

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the SSO token.

     > **Important**
     >
     > If you are configuring an SP connection to bridge one or more identity providers to a service provider, consider mapping the original issuer of the assertions into an attribute by selecting **Context** as the source and **Authenticating Authority** as the value. This is important when bridging multiple identity providers to one service provider, where the service provider should take the information about the original issuer into consideration before granting access to protected resources.
     > For more information, see Bridging multiple IdPs to an SP.

> **Note**
>
> Because the **HTTP Request** context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values (see **Expression**).

- **LDAP/JDBC/Other** (when a datastore is used)

  Values are returned from your datastore (if used). When you make this selection, the Value list is populated by the attributes from the datastore.

- **Expression** (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see Attribute mapping expressions). All of the variables available for text entries (see below) are also available for expressions.

- **No Mapping**

  Select this option to ignore the **Value** field, causing no value selection to be necessary.

- **Text**

  This can be text only, or you can mix text with references to any of the values from your user-datastore using this syntax:

  `[.codeph]``$\{ds.attr-source-id.attribute}```

  where `attr-source-id` is the **Attribute Source ID** value (see Choosing a datastore for Attribute Query) and `attribute` is any of the datastore attributes you have selected.

  There are a variety of reasons why you might hard code a text value. For example, if your SP's web application provides a service based on your company's name, you might provide that attribute value as a constant.

  > **Tip**
  >
  > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Next** to save changes.

Defining issuance criteria for Attribute Query

Use the **Issuance Criteria** tab to define issuance criteria for the service provider (SP) Attribute Query profile.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.

2. In the **Source** list, select the attribute's source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

+

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** As the **HTTP Request** context value is retrieved as a Java object rather than text, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore	Select to evaluate attributes returned from a datastore, if configured.
Mapped Attributes	Select to evaluate the mapped attributes.

1. In the **Attribute Name** list, select the attribute to be evaluated.

2. In the **Condition** list, select the comparison method.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**

- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> ⓘ **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> ⓘ **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ The **Error Result** field is used by the `StatusMessage` element in the SAML response to the SP.

+ Using an error code in the **Error Result** field allows an application to process the code in a variety of ways; for example, display an error message or e-mail an administrator.

1. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > ⓘ **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.

6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Specifying security policy

The **Specify Security Policy** tab allows you to specify the digital signing and encryption policy to which you and your partner have agreed.

About this task

> **Note**
>
> The selections you make on this tab will trigger requirements for setting up Credentials. For more information, see Configuring credentials.

Steps

1. Select or clear the check boxes.
2. Click **Next** or **Done**.

Reviewing the Attribute Query configuration

Review and save your Attribute Query configuration changes on the **Summary** tab.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Configuring credentials

The **Credentials** tab provides the launching point for configuring security requirements you might need, depending on the federation protocol you are using and the choices you make.

Steps

- To continue, click **Configure Credentials**.

See subsequent topics for configuration steps.

Configuring back-channel authentication (SAML)

Depending on your browser single sign-on (SSO) use cases, the administrative console prompts you to configure authentication requirements for inbound messages, outbound messages, or both.

About this task

See the following table for more information about the back-channel configuration (SAML) authentication requirements.

Use case	Back-channel authentication requirements	Back-channel messages
A connection is configured with a SAML ACS endpoint that uses the artifact binding on **Protocol Settings > Assertion Consumer Service URL**.	Inbound	Artifact resolution requests
A connection is configured with a SAML 2.0 SLO endpoint that uses the artifact binding on **Protocol Settings > SLO Service URLs**.	Inbound	Artifact resolution requests SOAP messages
A connection is configured with a SAML 2.0 SLO endpoint that uses the SOAP binding on **Protocol Settings > SLO Service URLs**.	Outbound	SOAP SLO messages
The SAML 2.0 Artifact binding is enabled on **Protocol Settings > Allowable SAML Bindings**.	Outbound	Outbound artifact resolution requests
The SOAP binding is enabled on **Protocol Settings > Allowable SAML Bindings**.	Inbound	Inbound SOAP messages
The SAML 2.0 Attribute Query profile is enabled on the **Connection Options** tab.	Inbound	Inbound Attribute Query requests

Steps

- See subsequent topics for configuration steps.

Configuring authentication requirements for outbound messages

You can configure the authentication requirements used to validate outbound messages in PingFederate.

Steps

1. On the **Back-Channel Authentication** tab, in the **Send to your partner** section, click **Configure**.
2. On the **Outbound SOAP Authentication Type** tab, choose one or more authentication methods.

HTTP Basic

When selected, the administrative console prompts you to enter the credentials on the **Basic SOAP Authentication (Outbound)** tab.You must obtain these credentials from your partner.

SSL Client Certificate

Applicable only if you specify an endpoint that uses HTTPS.When selected, the administrative console prompts you to specify your client certificate on the **SSL Authentication Certificate** tab. If you have not yet created or imported the client certificate, click **Manage Certificates** to do so. For more information, see Manage SSL client keys and certificates.

> **Important**
>
> When exporting this client certificate for your partner, choose the **Certificate Only** option.

Digital Signature (Browser SSO profile only)

You select a signing certificate on the **Digital Signature Settings** tab.This option leverages on the digital signature of the message.

Perform validation on partner's SSL server certificate when SSL used

By default, PingFederate validates your partner's HTTPS server certificate, verifying that the certificate chain is rooted by a trusted certificate authority (CA) and that the hostname matches the certificate's common name (CN).Clear the associated check box if you do not want this validation to occur.

These options can be used in any combination or independently.

3. On the **Summary** tab, review your configuration and perform one of the following tasks.

Amend your configuration

Click the corresponding tab title and then follow the configuration wizard to complete the task.

Keep your changes

Click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Configuring authentication requirements for inbound messages

You can configure the authentication requirements used to validate inbound messages in PingFederate.

Steps

1. On the **Back-Channel Authentication** tab, in the **Received from your partner** section, click **Configure**.
2. On the **Inbound Authentication Type** tab, choose one or more authentication methods.

   ### *HTTP Basic*

   When selected, the administrative console prompts you to enter the credentials on the **Basic SOAP Authentication (Inbound)** tab.

   > **Important**
   >
   > If you are configuring more than one connection that uses the artifact or HTTP profile, you must ensure that the username is unique for each connection. You must communicate these credentials to your partner out-of-band.

   ### *SSL Client Certificate*

   When selected, the administrative console prompts you to specify the trust model and the related certificate settings on subsequent windows. See the next step.

   ### *Digital Signature (Browser SSO profile only)*

   You select a signing certificate on the **Signature Verification Settings** tab.This option leverages on the digital signature of the message.

   ### *Require SSL*

   When selected, incoming HTTP transmissions must use a secure channel. This option is selected by default.You can clear the check box if you do not require a secure channel and client certificate authentication.

   For SAML 2.0, use these options in any combination or independently. For SAML 1.x, you must enable HTTP Basic authentication, client certificate authentication, or both. You can also add digital signing to ensure message integrity.
3. If you chose **SSL Client Certificate** in the previous step, select a trust model on the **Certificate Verification Method** tab.

   ### *Anchored*

   The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you can also restrict the issuer to a specific Trusted CA to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections. The CA's certificate must be imported into the PingFederate Trusted CA store on the **Trusted CAs**window..

   ### *Unanchored*

   The partner certificate is self-signed or you want to trust a specified certificate.

> **Note**
>
> When anchored certificates are used between partners, certificates can be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated. For more information, see Digital signing policy coordination.

Trust model	Subsequent steps
Anchored	On the **Subject DN** tab: 1. Enter the `Subject D`N of the certificate. 2. Optionally, select the **Restrict Issuer** check box and enter the `Issuer DN` of the certificate. **Important** Consider enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.
Unanchored	On the **SSL Verification Certificate** tab, select the client certification from your partner. If you have not yet imported the client certificate from your partner, click **Manage Certificates** to do so. For more information, see Managing certificates from partners.

4. On the **Summary** tab, review your configuration and perform one of the following tasks.

Amend your configuration

Click the corresponding tab title and then follow the configuration wizard to complete the task.

Keep your changes

Click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Configuring digital signatures for service provider connections

Digital signing is required for browser-based single sign-on (SSO) tokens and single logout (SLO) messages sent through POST or redirect bindings.

About this task

Digital signing is also required for WS-Trust STS service provider (SP) connections, for signing the outbound SAML security tokens.

> **Note**
>
> Configuring digital signatures for SP connections is just one step in configuring an SP connection. For more information, see SP connection management.

For browser-based SSO, digital signing is not always required for profiles using the artifact or SOAP bindings unless you chose to sign the SAML assertion on **Protocol Settings > Signature Policy**, or the artifact resolution messages on **Back-Channel Authentication > Outbound SOAP Authentication Type**.

If digital signing is not required, PingFederate does not show the **Digital Signature Settings** tab.

Steps

1. On the **Digital Signature Settings** tab, select the certificate that you will use to sign the SSO tokens and SLO messages for the SP.

2. Select a signing certificate from the **Signing Certificate** list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates**. For more information, see Manage digital signing certificates and decryption keys.

   > **Note**
   >
   > For WS-Federation connections using JSON Web Tokens (JWTs), only EC and RSA certificates are supported. RSA certificates must have a minimum key size of 2,048 bits. The **Signing Certificate** list automatically filters out certificates that do not meet these requirements.

3. **Optional:** Select a **Secondary Signing Certificate** for inclusion in the connection metadata.

   > **Note**
   >
   > You can't add a secondary certificate if the primary certificate has certificate rotation enabled. Also, you can't use a certificate that has rotation enabled as the secondary signing certificate.

   > **Tip**
   >
   > To deselect an existing secondary signing certificate, select **-SELECT-** in the **Secondary Signing Certificate** list. You can then delete the certificate by clicking **Manage Certificates** and selecting **Delete** from the **Actions** list.

4. **Optional:** Select the **Include the certificate in the signature <KeyInfo> element** check box if you have agreed to send your public key with the message.

   > **Note**
   >
   > For WS-Trust STS, the `<KeyInfo>` element in the SAML token includes a reference to the certificate rather than the full certificate by default unless this check box is checked.

   > **Note**
   >
   > This step is not applicable to WS-Federation connections using JWTs.

   Select the **Include the raw key in the signature <KeyValue> element** check box if your partner agreement requires it.

   Select the signing algorithm from the list.

The default selection is **RSA SHA256** or **ECDSA SHA256**, depending on the **Key Algorithm** value of the selected digital signing certificate. Make a different selection if you and your partner have agreed to use a stronger algorithm. For a list of the available signing algorithms and their URIs, see Signing algorithms.

Configuring signature verification settings (SAML 2.0)

You can configure the signature verification settings for the certificates in the PingFederate administrative console.

About this task

Depending on your partner agreement, digital signature processing might be required.

If you choose to require digital signatures on SAML 2.0 authentication requests on **Protocol Settings > Signature Policy** or inbound messages on **Back-Channel Authentication > Inbound Authentication Type**, you must configure the required certificate information that PingFederate can use to verify the signed messages.

The **Signature Verification Settings** tab is the launching point for this task. If digital signature verification is not required, the **Signature Verification Settings** tab is not shown.

Steps

1. On the **Signature Verification Settings** tab, click **Manage Signature Verification Settings**.
2. On the **Trust Model** window, select a trust model on the **Certificate Verification Method** tab.

Anchored

The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you can also restrict the issuer to a specific Trusted CA to mitigate potential man-in-the-middle attacks and provide a means to isolate certificates used by different connections. The CA's certificate must be imported into the PingFederate Trusted CA store on **Security > Certificate & Key Management > Trusted CAs**.

> **Important**
>
> If you are using the redirect binding for single logout (SLO), you cannot use anchored certificates because SAML 2.0 does not permit certificates to be included using this transport method.

Unanchored

The partner certificate is self-signed or you want to trust a specified certificate.

> **Note**
>
> When anchored certificates are used between partners, certificates can be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated. For more information, see Digital signing policy coordination.

Trust model	Subsequent steps
Anchored	On the **Subject DN** tab: 1. Enter the `Subject DN` of the certificate or extract it from your service provider (SP) partner's certificate if the certificate is stored on an accessible file system. 2. Optionally, select the **Restrict Issuer** check box and enter the `Issuer DN` of the certificate. Alternatively, extract it from your partner's certificate. **Important** You can enable this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.
Unanchored	On the **Signature Verification Certificate** tab: 1. Select a certificate from the list. If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so. For more information, see Managing certificates from partners. 2. Optionally, select additional certificates. **Note** When configured, PingFederate considers a digital signature valid so long as it can verify the signature using one of the certificates from this list. **Tip** This is useful in situations where your partner has sent you a certificate to replace the current certificate. Adding this second certificate allows PingFederate to continue validating digital signatures as the partner switches to the new signing certificate. It also adds support for scenarios where your partner uses a pool for certificates to sign its messages. Adding these certificates ensures digital signatures can be validated as the partner rotates its signing certificates.

3. On the **Summary** tab, review your configuration and perform one of the following tasks.

Amend your configuration

Click the corresponding tab title and then follow the configuration wizard to complete the task.

Keep your changes

Click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Selecting an encryption certificate

For browser-based single sign-on (SSO), if you choose to encrypt all or part of an SSO assertion on **Protocol Settings > Encryption Policy**, you must identify the certificate that PingFederate can use to do so.

About this task

You must also select a certificate if your requirements include encrypting an assertion in response to an attribute query on **Attribute Query > Security Policy**.

For WS-Trust security token service (STS), this configuration is also required if you enabled the **Generate Key for SAML Holder of Key Subject Confirmation Method** or **Encrypt SAML 2.0 Assertion** option, or both, on **WS-Trust > Protocol Settings**.

If encryption is not required, the **Select XML Encryption Certificate** tab is not shown.

Steps

1. **Optional:** Select an option under **Block Encryption Algorithm**.

   > **Important**
   >
   > Because of the import restrictions of some countries, Oracle Server Java SE Runtime Environment (JRE) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, enable the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy. For more information, see the Java 8 release notes in Oracle’s documentation.
   > For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide in Oracle’s documentation.

   The default selection is **AES-128**.

   For more information about XML block encryption and key transport algorithms, see XML Encryption Syntax and Processing from W3C.

2. Select an option under **Key Transport Algorithm**.

   > **Note**
   >
   > Due to security risks associated with the RSA-v1.5 algorithm used for key transport, it is no longer available for new connections. Existing connections in which this algorithm is configured continue to support it. However, you should upgrade such connections to use a newer algorithm.

   The default selection is **RSA-OAEP**.

3. Select a partner certificate from the list.

   If you have not imported the certificate from your partner, click **Manage Certificates** to do so. For more information see Managing certificates from partners.

Selecting a decryption key (SAML 2.0)

To enable inbound encryption in PingFederate, you must select a certificate on the decryption key.

About this task

When you choose to encrypt the name identifier (`SAML_SUBJECT`) on **Protocol Settings > Encryption Policy**, you can also allow the service provider (SP) to encrypt the name identifier in its single logout (SLO) requests, if the SP-initiated single sign-on (SSO) profile is enabled for the connection. To enable this inbound encryption, you must specify at least one certificate on the **Select Decryption Keys** tab.

If decryption is not required, the **Select Decryption Keys** window is not shown.

Steps

1. Select the primary XML decryption key from the list.

   If you have not created or imported your certificate into PingFederate, click **Manage Certificates**. For more information, see Manage digital signing certificates and decryption keys.

2. **Optional:** Select the secondary XML decryption key from the list.

Reviewing SP credential settings

You can review, modify, save, and discard changes to your service provider (SP) credential settings.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Configuring outbound provisioning

PingFederate's outbound provisioning allows an identity provider (IdP) to create and maintain user accounts at standards-based partner sites using System for Cross-domain Identity Management (SCIM) as well as select-proprietary provisioning partner sites that are protocol-enabled.

About this task

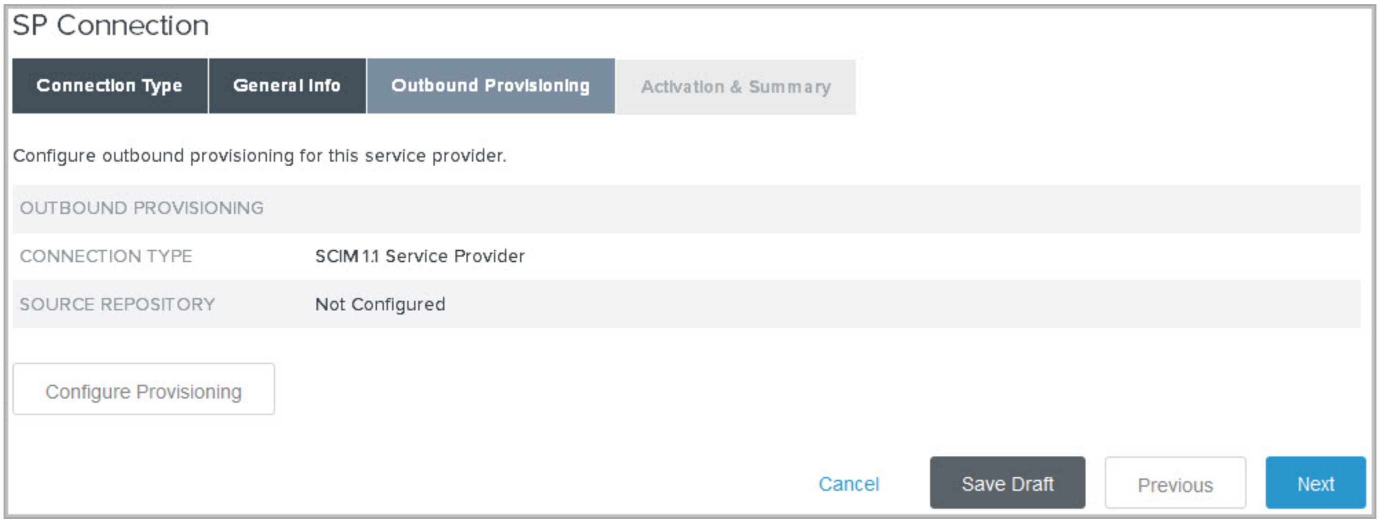
In the **SP Connections** window, configure outbound provisioning. For more information, see Outbound provisioning for IdPs.

Note

This configuration task is presented in the administrative console only when you enable the **Outbound Provisioning** protocol. For more information, see Choosing an SP connection type.

Steps

- Go to **Applications** > **Integration** > **SP Connection**. To continue, click **Configure Provisioning**.



SP Connection

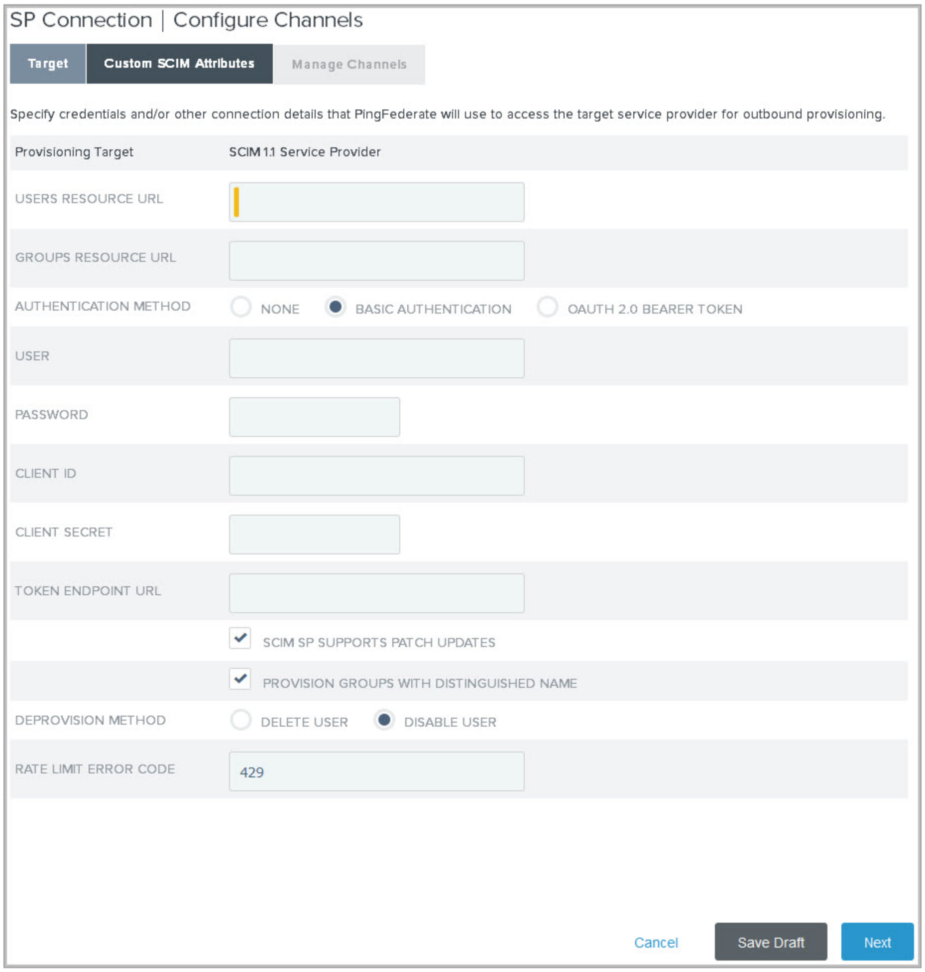
Configure outbound provisioning for this service provider.

OUTBOUND PROVISIONING	
CONNECTION TYPE	SCIM 1.1 Service Provider
SOURCE REPOSITORY	Not Configured

Defining a provisioning target

You can define a provisioning target, including the provider's web-service endpoint for provisioning users, and, if required, credentials that PingFederate uses for authentication to the provisioning API for the service provider (SP).

About this task



SP Connection | Configure Channels

Specify credentials and/or other connection details that PingFederate will use to access the target service provider for outbound provisioning.

Provisioning Target	SCIM 1.1 Service Provider
USERS RESOURCE URL	
GROUPS RESOURCE URL	
AUTHENTICATION METHOD	○ NONE ● BASIC AUTHENTICATION ○ OAUTH 2.0 BEARER TOKEN
USER	
PASSWORD	
CLIENT ID	
CLIENT SECRET	
TOKEN ENDPOINT URL	
	☑ SCIM SP SUPPORTS PATCH UPDATES
	☑ PROVISION GROUPS WITH DISTINGUISHED NAME
DEPROVISION METHOD	○ DELETE USER ● DISABLE USER
RATE LIMIT ERROR CODE	429

Note

The target configuration settings vary among System for Cross-domain Identity Management (SCIM) outbound provisioning and various Software as a Service (SaaS) provisioning.

For SCIM provisioning to PingOne for Enterprise, sign on to the PingOne admin portal and review the target information on the **Setup > Identity Repository** tab.

For any SaaS Connector target, refer to the documentation in the add-on distribution package.

The following steps describe the fields required for the bundled PingFederate provisioning plugin for SCIM partners.

Steps

1. Enter the endpoint for managing users in the **Users Resource URL** field. For example, https://example.com/v1/Users

   This field is always required for SCIM outbound provisioning.

2. Go to **Applications** > **Integration** > **SP Connections** > **SP Connection** > **Configure Channels**.

3. On the **Custom SCIM Attributes** tab, configure the remaining outbound provisioning settings.

   Refer to the following table for detailed information about each field:

Field	Description
Groups Resource URL	The partner's group management endpoint. For example, https://example.com/v1/Groups Required if the partner supports this notion and groups should be provisioned.
Authentication Method	The authentication scheme that the partner's endpoints support. Available options: ◦ **None** ◦ **Basic Authentication** (default) ◦ **OAuth 2.0 Bearer Token**: Uses the resource owner grant type by submitting the client ID, client secret, username, and password to the configured token endpoint URL in exchange for an access token that will be sent in each SCIM request.
User **Password**	Valid credentials to access the partner's endpoint. Required if **Basic Authentication** is the selected authentication method.
Client ID **Client Secret** **Token Endpoint URL**	Valid OAuth client credentials and token endpoint to access the partner's endpoint. Required if **OAuth 2.0 Bearer Token** is the selected authentication method.
SCIM SP Supports Patch Updates	Clear this checkbox if the partner does not support PATCH updates. Learn more about PATCH in the SCIM specification. This checkbox is selected by default.
Provision Groups with Distinguished Name	Select this checkbox to provision groups by supplying complete LDAP distinguished names (DNs) rather than only common names (CNs) to identify groups. Some SCIM partners, including PingOne for Enterprise, allow administrators to parse full DNs when necessary, such as in the case of duplicate CNs, to determine group access mapping to specific applications based on other DN elements. Consult the partner for its requirement. This checkbox is selected by default.
Deprovision Method	Deprovisioning is triggered when previously provisioned users no longer meet the condition set in **Manage Channels > Channel > Source Location**. Available options: ◦ **Disable User** (default) This option deactivates the user accounts. ◦ **Delete User** This option removes the user accounts. **Note** For SaaS provisioning, the provisioner does not necessarily remove deprovisioned users from target data stores in accordance with common practice. Instead, their status changes to indicate that the accounts are no longer active.
Rate Limit Error Code	The expected error code returned by the partner based on its rate-limiting threshold. The default value is `429`.

4. Click **Next**.

> **Note**
>
> For some provisioning plugins, including the built-in SCIM outbound provisioner, when you enter or change credentials and click **Next**, PingFederate immediately tests connectivity to the target.

Specifying custom SCIM attributes

You can configure simple, multivalued, and complex custom System for Cross-domain Identity Management (SCIM) attributes in PingFederate.

About this task

PingFederate supports SCIM attributes in the core schema and custom attributes through a schema extension.

> **Note**
>
> Custom attributes are optional. If your use case does not require any additional attributes, click **Next** on the **Custom SCIM Attributes** tab.

To support custom attributes, you must specify the schema extension and the custom attributes in the connection. There are four attribute types:

- Simple attributes
- Simple multivalued attributes
- Complex attributes
- Complex multivalued attributesThe following fragment illustrates a SCIM message supporting schema extension `urn:scim:schemas:extension:custom:1.0` with four attributes, one of each attribute type. The table afterward describes the details of each attribute.

```
{
  "userName":"CBrown",
  "active":true,
  "schemas":[
    "urn:scim:schemas:core:1.0",
    "urn:scim:schemas:extension:custom:1.0"
  ],
  ...
  "urn:scim:schemas:extension:custom:1.0":{
    "supervisor":"JSmith",
    "territories":[
      "Montana",
      "Idaho",
      "Wyoming"
    ],
    "options":{
      "quantity":"10000",
      "strike"  :"5.25",
      "first"   :"2017-12-01",
      "last"    :"2025-03-31"
    },
    "tablets":[
      {
        "model" :"8086",
        "serial":"5500-2020-965",
        "type"  :"office"
      },
      {
        "model" :"8088",
        "serial":"5500-2040-151",
        "type"  :"remote"
      }
    ]
  }
}
```

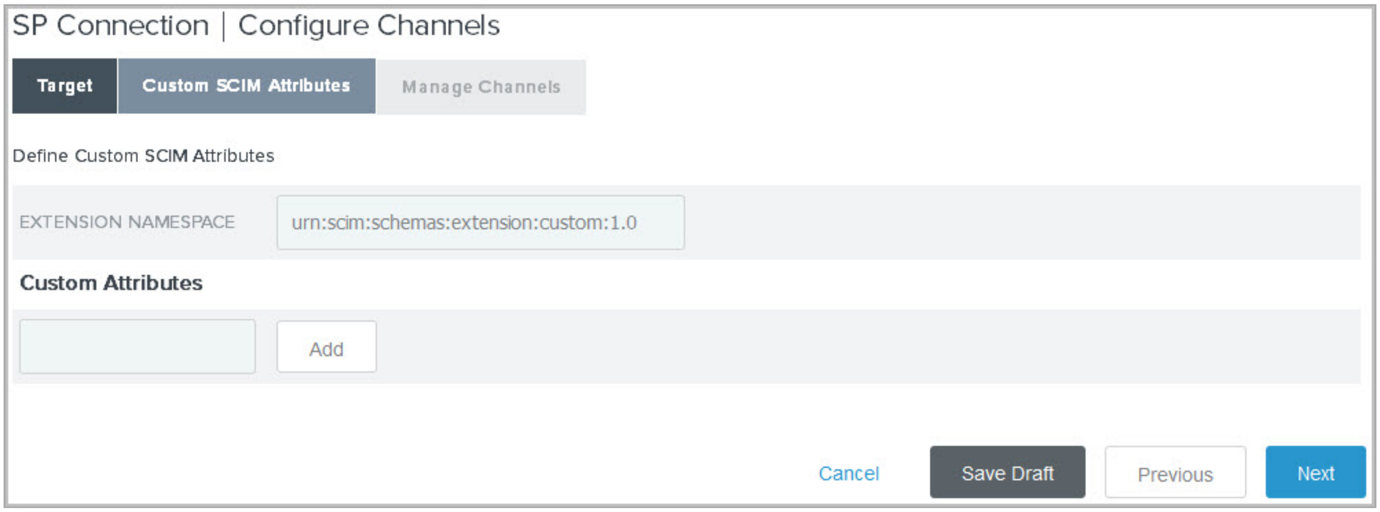
Attribute Name	Attribute Type	Sub-Attributes (Complex)
`supervisor`	Simple	Not applicable
`territories`	Simple multivalued	Not applicable
`options`	Complex	`quantity`, `strike`, `first`, and `last`
`tablets`	Complex multivalued	`model`, `serial`, and `type`. **Note** `type` is a reserved sub-attribute for a complex multivalued attribute.

Tip

For more information about SCIM and attribute types, see the website www.simplecloud.info.

Steps

1. Go to **Applications** > **Integration** > **SP Connection** > **Configure Channels**. Specify the URI of the schema extension in the **Extension Namespace** field.



SP Connection | Configure Channels

Define Custom SCIM Attributes

EXTENSION NAMESPACE: urn:scim:schemas:extension:custom:1.0

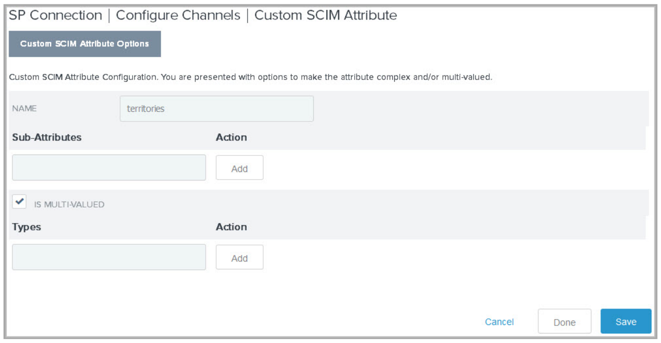
Custom Attributes

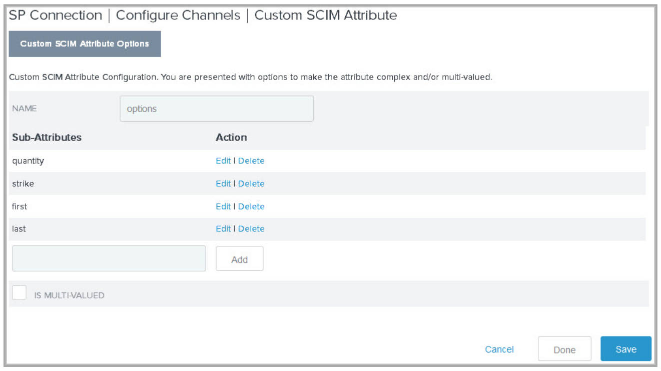
> **Tip**
>
> The default value is `urn:scim:schemas:extension:custom:1.0` . You can keep this value if your partner identifies custom attributes by this URI in its SCIM messages.

2. Enter an attribute name and click **Add** to add a custom attribute.Repeat this step to add more custom attributes as needed.

> **Tip**
>
> Use the **Delete** and **Undelete** workflow to remove or cancel the removal request of existing custom attributes.

3. Click **Edit** next to the custom attribute to perform one of the following tasks.

Change the attribute name	Replace the current value in the Name field.Click Done. Screen capture illustrating a simple attribute on the Custom SCIM Attribute Options tab.
Set the attribute as a simple multivalued attribute	1. Select the **Is Multivalued** check box. 2. Click **Done**.  SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute Options Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued. NAME: territories Sub-Attributes / Action: Add ☑ IS MULTI-VALUED Types / Action: Add Cancel Done Save

Change the attribute name	Replace the current value in the Name field.Click Done. Screen capture illustrating a simple attribute on the Custom SCIM Attribute Options tab.
Add sub-attributes to make the attribute a complex attribute	1. Enter a sub-attribute and click **Add**. Repeat this step to add more sub-attributes as needed. 2. Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the name of a sub-attribute. Use the **Delete** and **Undelete** workflow to remove a sub-attribute or cancel the removal request. 3. Click **Done**.  SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued. NAME options Sub-Attributes Action quantity Edit I Delete strike Edit I Delete first Edit I Delete last Edit I Delete □ IS MULTI-VALUED Custom SCIM Attribute Options Add Cancel Done Save

<table>
<tr><th>Change the attribute name</th><th>Replace the current value in the Name field.Click Done. Screen capture illustrating a simple attribute on the Custom SCIM Attribute Options tab.</th></tr>
<tr><td>Add sub-attributes and set the attribute as a complex multivalued attribute</td><td>

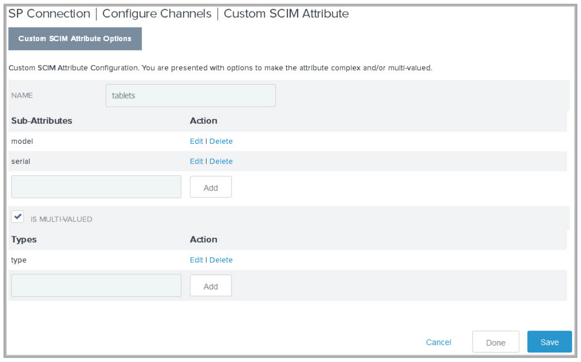
1. Enter a sub-attribute and click **Add**. Repeat this step to add more sub-attributes as needed.

   > **Tip**
   > Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the name of a sub-attribute. Use the **Delete** and **Undelete** workflow to remove a sub-attribute or cancel the removal request.

2. Select the **Is Multivalued** check box.
3. Specify at least one value under the **Types** column for `type`, a reserved sub-attribute for a complex multivalued attribute.

   > **Tip**
   > Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the `type` value. Use the **Delete** and **Undelete** workflow to remove a `type` value or cancel the removal request.

4. Click **Done**.



SP Connection | Configure Channels | Custom SCIM Attribute

Custom SCIM Attribute Options

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME tablets

Sub-Attributes	Action
model	Edit I Delete
serial	Edit I Delete
	Add

☑ IS MULTI-VALUED

Types	Action
type	Edit I Delete
	Add

Cancel Done Save

</td></tr>
</table>

Managing channels

You can manage provisioning channels in the PingFederate administrative console to add, modify, and delete channels.

About this task

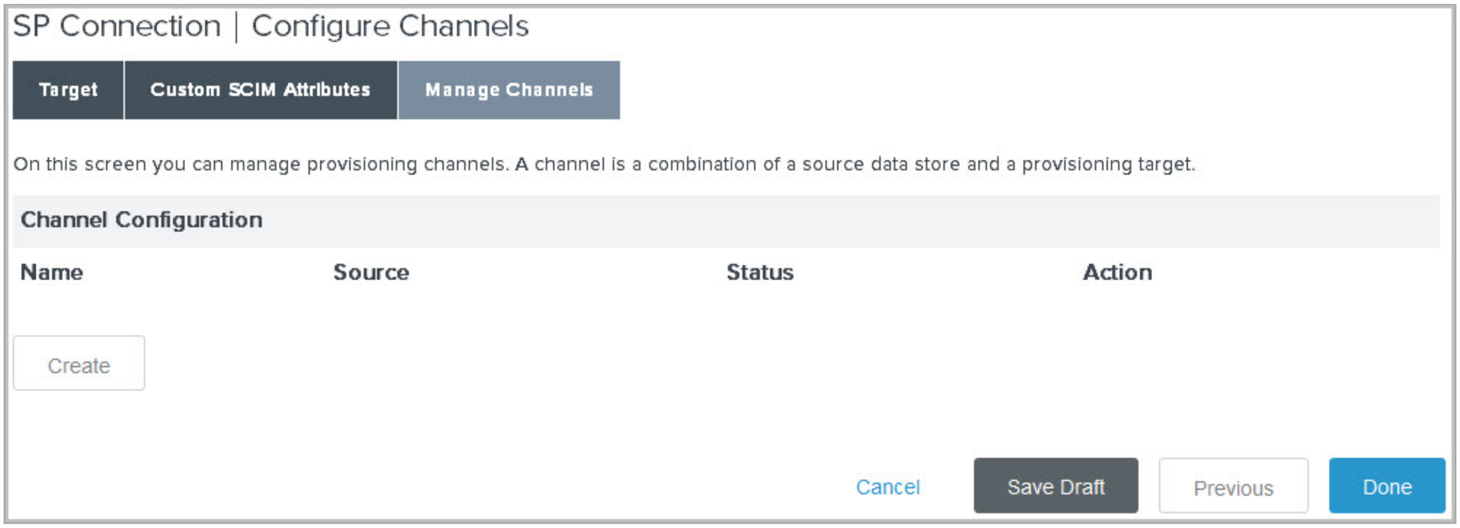
A provisioning channel is a mapping configuration between user attributes contained in a source user store and attributes supported or required by the targeted software-service application. You can have multiple channels to the same target as needed, such as if your organization has separate LDAP stores, or different nodes in the same store, for various user groups needing single sign-on (SSO) access and provisioning to the same domain.

Important

If two different channels have the same group name, the provisioning channel will overwrite the group members in the other channel. For example, channel1 reads from AD1 where user1 is in a group named "Admins," and channel2 reads from AD2 where user2 is in a group also named "Admins." When group membership is provisioned to the same target, channel2 will overwrite the data provisioned by channel1 for the "Admins" group so that user2 will now be in "Admins" in the target.

Tip

There can be only one provisioning target per connection. If your organization subscribes to multiple provisioning targets for which you need provisioning support, you need a separate service provider (SP) connection for each provisioning target.



SP Connection | Configure Channels

On this screen you can manage provisioning channels. A channel is a combination of a source data store and a provisioning target.

Channel Configuration

Name	Source	Status	Action

Go to **Applications** > **Integration** > **SP Connection** > **Configure Channels**. On the **Manage Channels** tab, you can perform the following tasks:

Steps

- To add a new channel, click **Create**.

  Alternatively, you can create a new channel by copying an existing channel and making other required changes.

- To modify existing channel settings, select an existing channel.
- To remove or cancel the removal request of the existing channel, use the **Delete** and **Undelete** workflow.

Specifying channel information

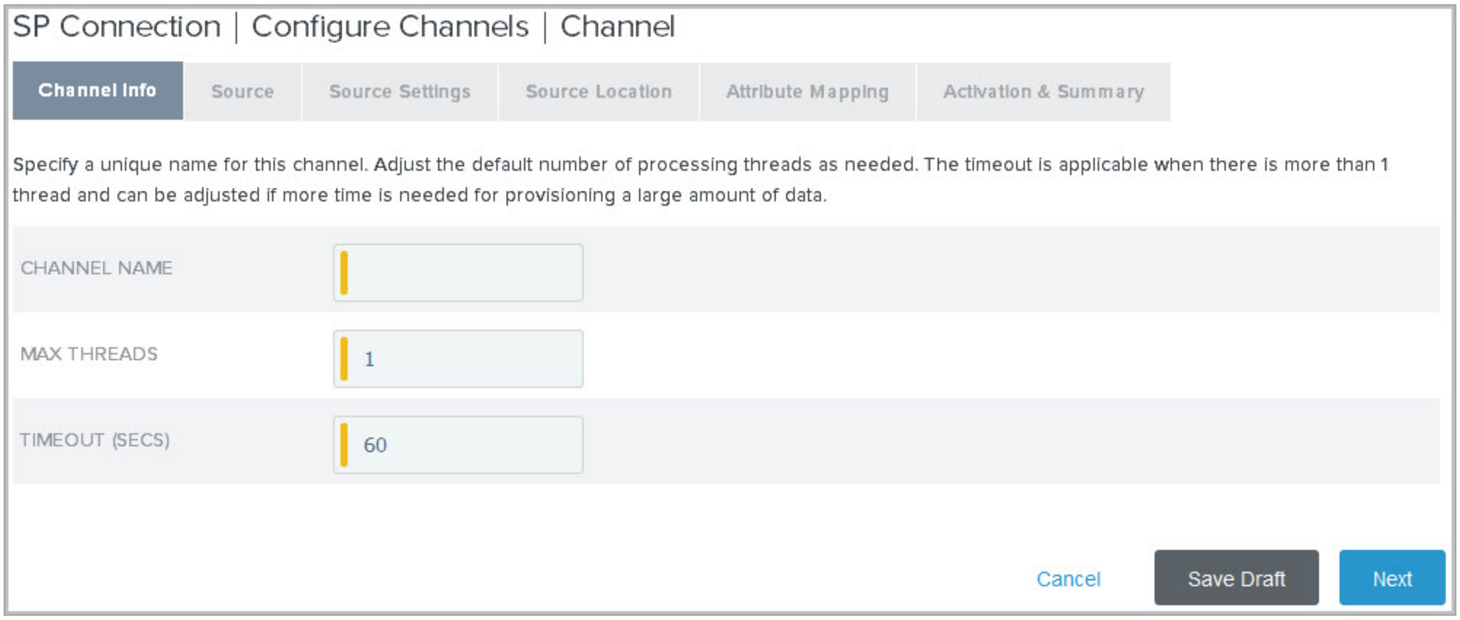
When configuring channels in PingFederate, you can specify channel information such as the channel name, max threads, and timeout.

About this task

On the **Channel Info** tab, specify a unique identifier for the channel and adjust the values for the **Max Threads** and **Timeout** fields as needed to optimize data-transfer performance, particularly if large numbers of records need to be provisioned at the target site. The **Max Threads** and **Timeout** settings apply to the phase of provisioning when PingFederate is pushing updates to the provisioning target.

The **Max Threads** setting determines how many threads PingFederate can use to send updates to the target. Using multiple threads should let PingFederate finish this provisioning phase faster. There is no particular order in which PingFederate processes updates. Each thread pulls the user and group update operations from a queue. As soon as a thread finishes one operation, it pulls another operation from the queue. Eventually there will be no more operations in the queue, so all threads will finish at close to the same time, but not exactly.

The **Timeout** setting determines how long PingFederate has to update a record. If PingFederate takes too long to update a record, it moves to the next record. During the next provisioning cycle, PingFederate will try again to update the skipped record.


SP Connection | Configure Channels | Channel

Specify a unique name for this channel. Adjust the default number of processing threads as needed. The timeout is applicable when there is more than 1 thread and can be adjusted if more time is needed for provisioning a large amount of data.

CHANNEL NAME	
MAX THREADS	1
TIMEOUT (SECS)	60

Steps

1. Go to **Applications** > **Integration** > **SP Connection** > **Configure Channels** > **Channel**. In the **Channel Name** field, enter a channel name.

   If you are copying a channel, you must enter a new value in this field.

2. **Optional:** Update the values for the **Max Threads** and **Timeout** fields.

   The **Timeout** value applies to individual user and group provisioning operations on the target service provider.

3. Click **Next**.

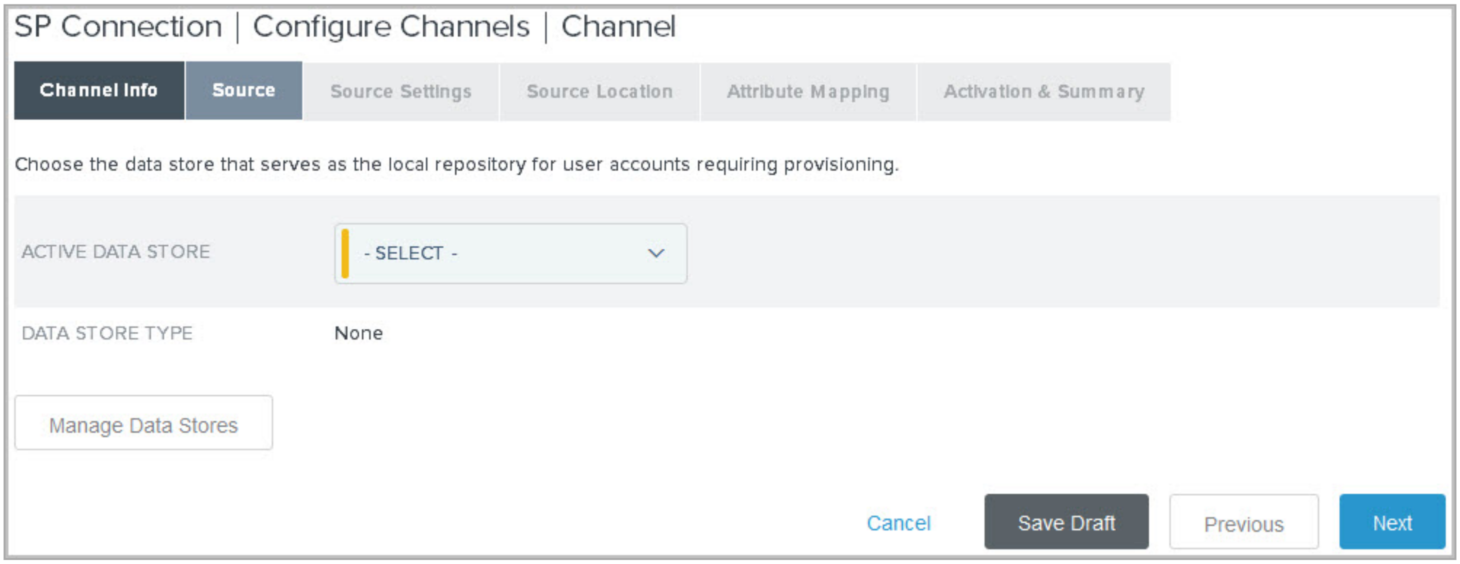
Identifying the source datastore

You can identify the source datastore for the service provider (SP) channel configuration in the PingFederate administrative console.

About this task

PingFederate supports PingDirectory, Microsoft Active Directory, Oracle Unified Directory, and Oracle Directory Server as source user repositories for outbound provisioning. However, you can use other types of LDAP servers, either identifying them as **Generic** or registering them with PingFederate. For more information, see the `sample.template.txt` in the `<pf_install>/pingfederate/server/default/conf/template/ldap-templates` directory.

Information from your user-datastore is used to supply mapped values for each user attribute required by the service provider (SP).



SP Connection | Configure Channels | Channel

Choose the data store that serves as the local repository for user accounts requiring provisioning.

ACTIVE DATA STORE: - SELECT -

DATA STORE TYPE: None

Steps

1. Go to **Applications** > **Integration** > **SP Connections** > **SP Connection** > **Configure Channels** > **Channel**.
2. On the **Source** tab, choose the LDAP store to use for this channel.

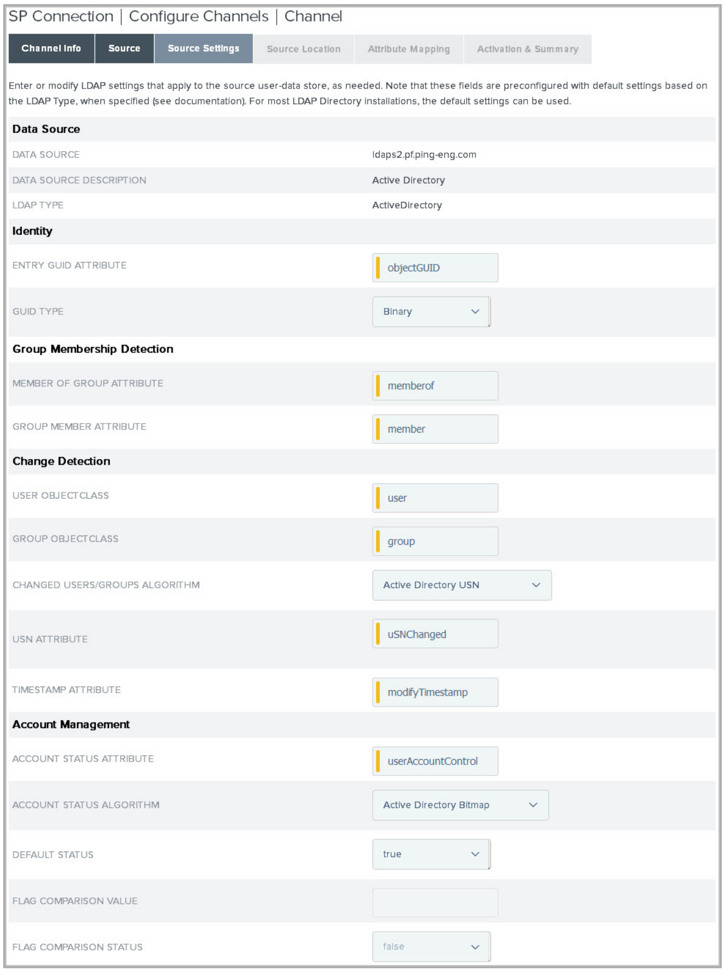
   If the datastore you want is not shown in the list, PingFederate is not configured to access the store. To create a connection to the datastore, click **Manage Data Stores**.
3. Click **Next**.

Modifying source settings

You can modify the source settings for the datastore configuration in the PingFederate administrative console. You can add, change, and remove user information.

About this task

The **Source Settings** tab shows the default configuration of the datastore selected on the **Source** tab, including settings used by the PingFederate provisioner to determine when user information is added, changed, or removed.



SP Connection | Configure Channels | Channel

Enter or modify LDAP settings that apply to the source user-data store, as needed. Note that these fields are preconfigured with default settings based on the LDAP Type, when specified (see documentation). For most LDAP Directory installations, the default settings can be used.

Data Source

DATA SOURCE: ldaps2.pf.ping-eng.com

DATA SOURCE DESCRIPTION: Active Directory

LDAP TYPE: ActiveDirectory

Identity

ENTRY GUID ATTRIBUTE: objectGUID

GUID TYPE: Binary

Group Membership Detection

MEMBER OF GROUP ATTRIBUTE: memberof

GROUP MEMBER ATTRIBUTE: member

Change Detection

USER OBJECTCLASS: user

GROUP OBJECTCLASS: group

CHANGED USERS/GROUPS ALGORITHM: Active Directory USN

USN ATTRIBUTE: uSNChanged

TIMESTAMP ATTRIBUTE: modifyTimestamp

Account Management

ACCOUNT STATUS ATTRIBUTE: userAccountControl

ACCOUNT STATUS ALGORITHM: Active Directory Bitmap

DEFAULT STATUS: true

FLAG COMPARISON VALUE:

FLAG COMPARISON STATUS: false

See the following table for more information about each field.

Field	Description
Entry GUID Attribute	The name of the attribute in the datastore representing the user's GUID.
GUID Type	Indicates whether the GUID is stored in binary or text format. Microsoft Active Directory is always binary. Other LDAP stores most often use text. ⓘ **Note** If binary is selected, ensure that the entered Entry GUID Attribute is also set as a binary attribute in the source LDAP datastore. For more information, see Setting advanced LDAP options.
Member of Group Attribute	A multivalued user attribute containing the distinguished names (DNs) of the groups to which an entry belongs. This attribute only applies to some LDAP servers, such as Microsoft Active Directory. When this attribute doesn't apply, the Group Member Attribute is used instead. Microsoft Active Directory use both values to provide a two-way mapping between user and group objects.

Field	Description
Group Member Attribute	The name of a multivalued group attribute used to track membership in the group using either DN or GUID values.
User objectClass	The LDAP object class to which user entries belong, used to restrict search results to user entries only. The default value is: • `inetOrgPerson` if the **Data Source** is PingDirectory • `person` if the **Data Source** is Oracle Directory Server or Oracle Unifier Directory • `objectGUID` if the **Data Source** is Microsoft Active Directory
Group objectClass	The LDAP object class to which group entries belong, used to restrict search results to group entries only.
Changed Users/Groups Algorithm	The method by which PingFederate determines if user records have been updated or new records added, thus requiring provisioning updates at the target site. The three choices are: • **Active Directory USN** – For Microsoft Active Directory only, this algorithm queries for update sequence numbers on user records that are larger than the last time records were checked. • **Timestamp** – Queries for timestamps on user records that are not older than the last time records were checked. This check is more efficient from the perspective of the PingFederate provisioner but can be more time consuming on the LDAP side, particularly with Oracle Unified Directory and Oracle Directory Server. • **Timestamp No Negation** – Queries for timestamps on user records that are newer than the last time records were checked. This algorithm is recommended for Oracle Unified Directory and Oracle Directory Server.
USN Attribute	The name of the attribute used to store the update sequence number. Applicable when the Microsoft Active Directory algorithm is chosen in the row above.
Timestamp Attribute	The name of the attribute used to store the timestamp on user records. ⓘ **Note** This attribute name is case-sensitive. Ensure the attribute name matches the name your directory uses. For example, in PingDirectory and Oracle, the attribute is `modifyTimestamp`, but in Microsoft Active Directory, the attribute is `modifyTimeStamp`. They have different capitalization.
Account Status Attribute	The name of the attribute in which the user's account status, active or inactive, is stored. For example, Microsoft Active Directory = `userAccountControl` and Oracle Directory Server = `nsaccountlock`.

Field	Description
Account Status Algorithm	The method by which PingFederate determines a user's account status. The values are: • **Active Directory Bitmap** for Microsoft Active Directory, which uses a bitmap for each user entry. For more information about `userAccountControl` flags, see Microsoft's knowledge base. • **Flag**– For Oracle Unified Directory, Oracle Directory Server, and other LDAP directories that use a separate attribute to store the user's status. When this option is selected, the Flag Comparison Value and Flag Comparison Status fields below are also used.
Default Status	Indicates the user's status if the attribute is missing.
Flag Comparison Value	Indicates the value for the attribute, such as `nsaccountlock`, that PingFederate expects to be returned. The value is case-sensitive. Used when the Account Status Algorithm is set to **Flag**.
Flag Comparison Status	Indicates whether the user is enabled or disabled when the flag has the value specified in the Flag Comparison Value field. Setting the value to `true` equals enabled, while setting the value to `false` equals disabled. For example, if the Account Status Attribute is set to `nsaccountlock`, and the Flag Comparison Value is set to `true`, and the Flag Comparison Status is set to `false`, then any users with `nsaccountlock=true` are disabled. Used when the Account Status Algorithm is set to **Flag**.

If you are using PingDirectory, Microsoft Active Directory, Oracle Unified Directory, or Oracle Directory Server, in most cases no changes are needed on this tab unless your datastore uses a customized schema.

If you are using a different LDAP directory, you must supply the required information on this tab unless you have defined a template for the datastore. For more information, see the `sample.template.txt` in the `<pf_install>/pingfederate/server/default/conf/template/ldap-templates` directory.

Steps

1. Modify the settings, as needed.

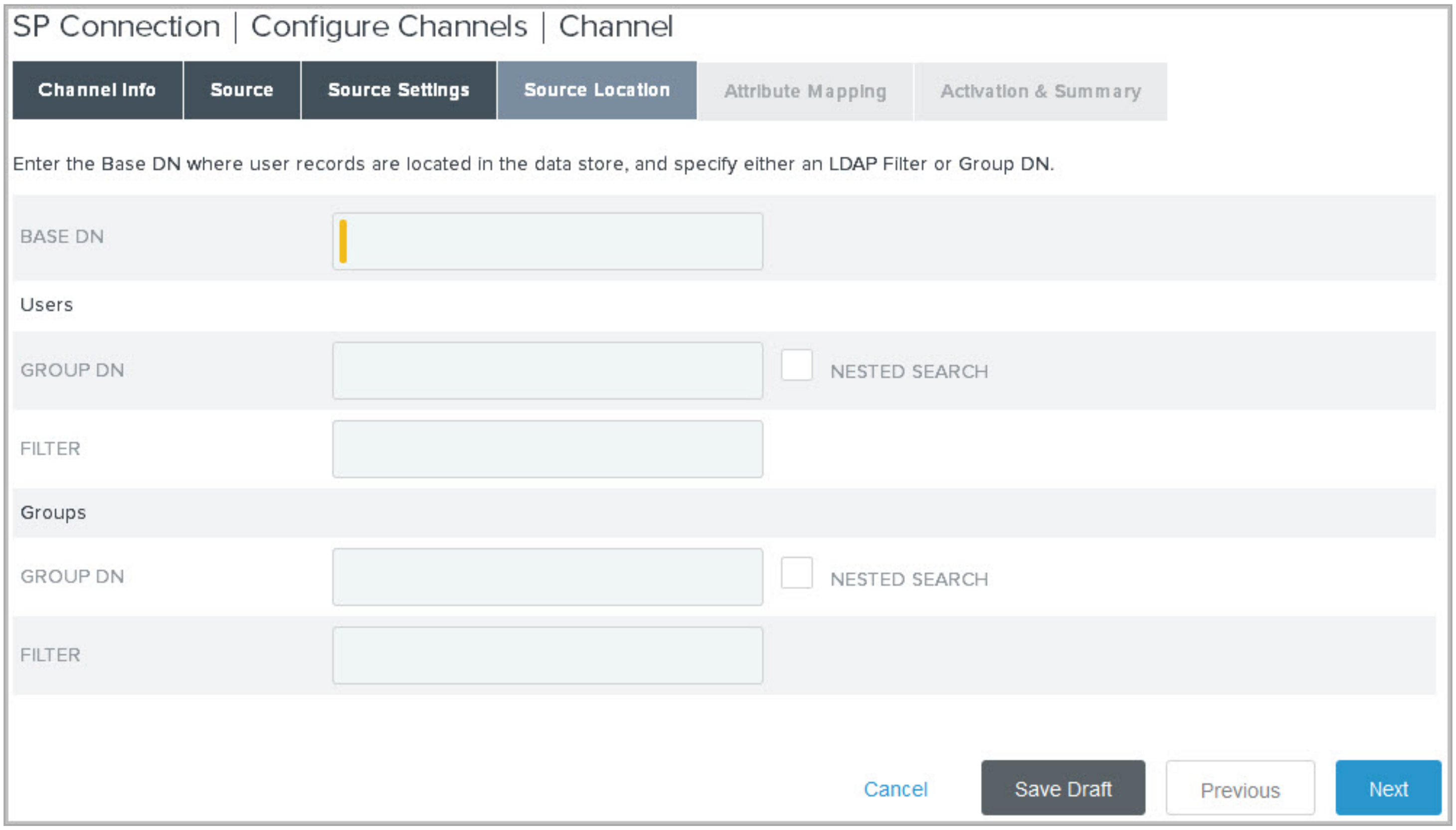
2. Click **Next**.

Specifying a source location

You can indicate on the **Source Location** tab where PingFederate should look for user records in the datastore.

About this task

The same location can be used to retrieve user-group distinguished names (DNs) for maintaining corresponding groups at the service provider (SP).



SP Connection | Configure Channels | Channel

Enter the Base DN where user records are located in the data store, and specify either an LDAP Filter or Group DN.

BASE DN

Users

GROUP DN □ NESTED SEARCH

FILTER

Groups

GROUP DN □ NESTED SEARCH

FILTER

After specifying the required base DN, you can provision users, and groups when applicable, based on group membership information or LDAP search results.

Note

Groups provisioning is supported for System for Cross-domain Identity Management (SCIM) and the Google Apps Connector (version 2.0 and higher) but might not be supported for other software as a service (SaaS) Connectors. If not, the associated fields under **Groups** on the **Source Location** tab are inactive. Support for the feature might become available in future SaaS Connector releases. See the documentation in your add-on distribution package.

Steps

1. Go to **Applications** > **Integration** > **SP Connections** > **Configure Channels** > **Channel**. In the **Base DN** field, enter the base DN where user records are stored.

   PingFederate looks only at this node level, or below it, for user accounts and groups (when applicable) that need to be provisioned based on the conditions set in the next step.

2. Specify group membership information or an LDAP filter to search for users, and groups when applicable, to be provisioned. For more information, see the following table.

Object	Field description
Users	**Group DN** The distinguished name (DN) of a group in the user repository whose member groups should be provisioned. Optionally, select the **Nested Search** check box to include users that are members of the specified group through nested group membership. Nested group membership is preserved for SCIM provisioning, and SaaS provisioning if the vendor and the SaaS Connectors support hierarchical structure in groups. ⓘ **Note** The **Nested Search** feature is available when PingDirectory, Microsoft Active Directory, Oracle Unified Directory, or Oracle Directory Server is selected as the source user repository. For more information, see Identifying the source datastore. **Filter** An LDAP search filter that returns user objects representing the users that should be provisioned. For information about LDAP filters, see your LDAP documentation. You might need to escape any special characters. **Important** The **Group DN** field is ignored when a **Filter** field value is configured. If you are using Active Directory, the filter must include `objectClass=user` for the provisioner to retrieve users.
Groups (when applicable)	**Group DN** The DN of the group in the user repository that should be provisioned. Optionally, select the **Nested Search** check box to include groups that are members of the specified group through nested group membership. Nested group membership is preserved for SCIM provisioning, and SaaS provisioning if the vendor and the SaaS Connectors support hierarchical structure in groups. ⓘ **Note** The **Nested Search** feature is available when PingDirectory, Microsoft Active Directory, Oracle Unified Directory, or Oracle Directory Server is selected as the source user repository. For more information, see Identifying the source datastore. **Filter** An LDAP search filter that returns group objects representing the groups that should be provisioned. For information about LDAP filters, refer to your LDAP documentation. You might need to escape any special characters. **Important** The **Group DN** field is ignored when a **Filter** field value is configured. If both the **Group DN** field and the **Filter** field are blank, no groups will be provisioned.

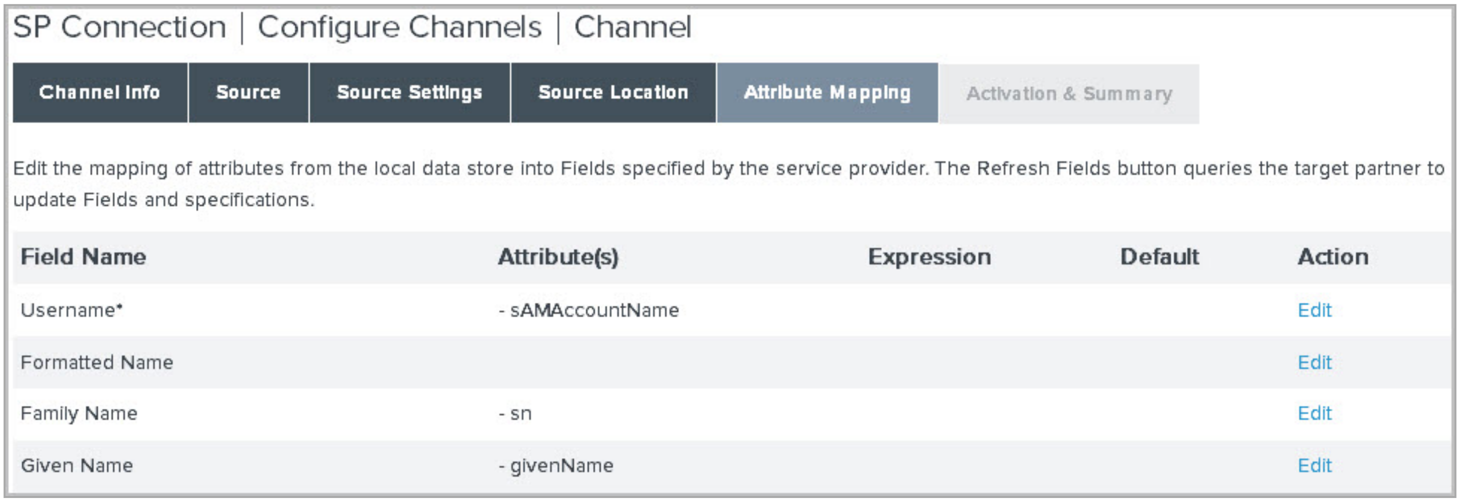
3. Click **Next**.

Mapping attributes

Mapping attributes determines how attributes from your user store are mapped to the System for Cross-domain Identity Management (SCIM) attributes in the core schema and custom attributes through a schema extension or to the provisioning fields supported for your organization’s software as a service (SaaS) customer account.

About this task

Edit the mapping of attributes from the local datastore into fields specified by the service provider (SP).



SP Connection | Configure Channels | Channel

Edit the mapping of attributes from the local data store into Fields specified by the service provider. The Refresh Fields button queries the target partner to update Fields and specifications.

Field Name	Attribute(s)	Expression	Default	Action
Username*	- sAMAccountName			Edit
Formatted Name				Edit
Family Name	- sn			Edit
Given Name	- givenName			Edit

> **Important**
>
> If you are provisioning for SCIM, your SP can make one or more optional core attributes mandatory. For more information, see the SCIM documentation from the SP or the SCIM Resource Schema representation.

> **Tip**
>
> For non-SCIM SaaS connectors, PingFederate automatically retrieves from the vendor the **Field Names** shown on this tab, but only on the first pass through the configuration flow. If you are using this configuration to modify an existing mapping configuration, click **Refresh Fields** to synchronize the list with the target if needed.

For each field, the **Attribute Mapping** option provides a means of adding or modifying the mapping details.

> **Note**
>
> All required attributes listed in the **Field Name** column, indicated with asterisks, must be mapped. Click **View Partner Field Specifications** for a summary of requirements for all fields specified for the target partner.
> For some fields, PingFederate preselects LDAP attributes commonly used to store the required values.

Steps

1. Go to **Applications > Integration > SP Connections** to open the **SP Connections** configuration window.

2. To edit an existing SP connection, open an SP connection by clicking on its name in the **Connection Name** column.

3. On the **Outbound Provisioning** tab, click **Configure Provisioning** to open the **Configure Channels** configuration window.

   > **Note**
   >
   > The **Outbound Provisioning** tab is only visible after you go to the **Connection Type** tab, select the **Outbound Provisioning** check box and in the **Type** list, select the type.

4. Go to the **Manage Channels** tab.

5. Select a channel.

6. Go to the **Attribute Mapping** tab.

7. To edit a field, click **Edit** in the **Action** column.

   > **Tip**
   >
   > If you have specified any custom attributes, they are listed at the end of the **Attribute Mapping** configuration.

8. On the **Attribute Mapping** tab, provide mapping details.

   > **Note**
   >
   > To prevent unexpected errors when reading or encoding binary attributes from Microsoft Active Directory (AD), add any AD binary attributes mapped during provisioning to the **LDAP Binary Attributes** list in the Data Store **Advanced LDAP Options**. For more information, see Setting advanced LDAP options.

9. Repeat steps for each attribute shown in the **Field Name** column as needed.

   > **Tip**
   >
   > For most fields, if you map more than one attribute from your datastore into a single field at the target location, then you must use an OGNL expression to indicate how to combine the attribute values.
   > The only exception is the `LDAP Attributes Map` field, which is provided primarily to support SCIM attributes specific to PingOne for Enterprise. This field can contain multiple attributes without using OGNL.

10. Click **Next**.

Specifying mapping details

Define specific mapping information for each field, required or optional, for provisioning as needed.

> **Caution**
>
> If end-users at your site are permitted to edit some of their own attributes directly in the LDAP store, ensure that the attributes are restricted and do not include any needed by the service provider to grant permissions.

Defining mapping information for a standard attribute

Before you begin

- Go to **Applications** > **Integration** > **SP Connections** to open the **SP Connections** configuration window.
- To edit an existing SP Connection, open an SP Connection by clicking on its name in the **Connection Name** column.
- On the **Outbound Provisioning** tab, click **Configure Provisioning** to open the **Configure Channels** configuration window.

> **Note**
>
> The **Outbound Provisioning** tab, is only visible after you select the **OutBound Provisioning** check box and the type in the **Type** list, on the **Connection Type** tab.

- Go to the **Manage Channels** tab.
- Click the name of the channel to edit it.

> **Tip**
>
> If you have specified any custom attributes, they are listed at the end of the **Attribute Mapping** configuration.

Steps

1. On the **Attribute Mapping** tab, click **Edit** in the **Action** column for the **Field Name** whose attributes you want to map.
2. Select the class containing a user-store attribute in the **Root Object Class** column that you want to map to the provisioning attribute shown in the **Field Name** column.

   > **Note**
   >
   > For some fields, you might not need to map specific user attributes. If so, supply a value in the **Default Value** field, skip this step, and go to step 5. For certain attributes, you can specify LDAP attributes and a default value, as needed.

3. Select the source attribute from the class in the **Attribute** column. Click **Add Attribute**.

   > **Note**
   >
   > If the selected source attribute is binary, ensure that the selected attribute is set as a binary attribute in the source LDAP datastore. For more information, see Setting advanced LDAP options.

4. Repeat the previous steps to add additional applicable attributes to use in a mapping expression.

   > **Note**
   >
   > You must add an attribute for it to be used in an expression.

5. **Optional:** If one or more attributes are specified: go to the **Value Definition** section, and in the **Default Value** field, enter or select a default value.

   If you have specified any custom attributes, they are listed at the end of the **Attribute Mapping** configuration.

   A list appears for this field if the vendor requires a choice among specified values. When an expression is also supplied, the default value is sent during provisioning if an error occurs when evaluating the expression.

6. If more than one attribute is used for mapping fields other than **LDAP Attributes Map**, in the **Value Definition** section, enter an expression.

    1. To create and validate the expression for the **Expression** field, click **Edit**.

7. Select one or more processing options.

Processing option	Description
Create Only	The field is provisioned only once and not subsequently updated. **Note** For SCIM, the `Password` attribute should be passed only when creating a user or updating the password. Select **Create Only** to limit when the `Password` attribute is passed.
Trim	Removes any white space from the attribute values.
Mask Log Values	Determines whether sensitive information, such as the `Password` attribute, will be masked in PingFederate log files.
Upper Case, Lower Case, or None	Transforms the attribute values to the case indicated unless the default, **None** option, is selected.
Parsing > Extract CN from DN	For attributes in the form of a distinguished name (DN), such as Group DNs in Active Directory, maps only the common name portion of the DN.
Parsing > Extract Username from Email	For attributes containing an email address, maps only the username.

8. Click **Done**.

Defining mapping information for a custom attribute

Steps

1. Select a sub-attribute in the **Attribute** column and list.

> **Note**
>
> Applicable only to complex attributes or complex multivalued attributes, see Specifying custom SCIM attributes.

2. Select the class containing a user-store attribute in the **Root Object Class** column that you want to map to the provisioning attribute shown in the **Field Name** column.

> **Note**
>
> For some fields, you might not need to map specific user attributes. If so, supply a value in the **Default Value** field, skip this step, and go to step 5. For certain attributes, you can specify both LDAP attributes and a default value, as needed.

3. Select the source attribute from the class in the **LDAP Attribute** column. Click **Add Attribute**.

> **Note**
>
> If the selected source attribute is binary, ensure that the selected attribute is set as a binary attribute in the source LDAP datastore. For more information, see Setting advanced LDAP options.

4. In the **Options** section, select one or more processing options.

Processing option	Description
Create Only	The field is provisioned only once and not subsequently updated. **Note** For System for Cross-domain identity Mangement (SCIM), the `Password` attribute should be passed only when creating a user or updating the password. Select **Create Only** to limit when the `Password` attribute is passed.
Trim	Removes any white space from the attribute values.
Mask Log Values	Determines whether sensitive information, such as the `Password` attribute, will be masked in PingFederate log files.
Upper Case, Lower Case, or None	Transforms the attribute values to the case indicated unless the **None** option is selected, the default.
Parsing > Extract CN from DN	For attributes in the form of a distinguished name (DN), such as Group DNs in Active Directory, maps only the common name portion of the DN.
Parsing > Extract Username from Email	For attributes containing an email address, maps only the username.

5. In the **Default Value** field, enter a default value.

6. Click **Add Mapping**.

> **Note**
>
> For complex attributes or complex multivalued attributes, repeat these steps to map additional sub-attributes as needed.

7. Click **Done**.

Reviewing channel settings

When you finish setting up a channel, you can choose to activate it immediately or activate the channel as needed.

About this task

On the **Activation & Summary** tab, review, activate, deactivate, or save your channel settings.

> **Note**
>
> A service provider connection must be active for any provisioning channels to be enabled. You can deactivate a channel at any time. When a channel is inactive, provisioning is suspended but single sign-on (SSO) and single logout (SLO) transactions can still occur if an associated connection is active.

Steps

1. Go to **Applications** > **Integration** > **SP Connections** > **SP Connection** > **Configure Channels** > **Channel**.
2. On the **Activation & Summary** tab, select **Active** or **Inactive** to toggle the status. Click **Save**.

   > **Caution**
   >
   > When a channel is activated, initial provisioning occurs as soon as the synchronization-frequency time period expires. See Configuring outbound provisioning settings. The default is 60 seconds. Because initial provisioning can consume considerable processing time, depending on the amount of data that needs to be transmitted, administrators should plan accordingly.

3. To modify channel settings, click the associated heading in the **Summary** column.

   > **Important**
   >
   > Click **Save** to save the channel configuration.

Reviewing SP connection settings

When you finish creating or modifying a service provider (SP) connection, you can review the connection settings and toggle the connection status.

About this task

On the **Activation & Summary** tab, you can review, amend, discard, or save your changes.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Result

> **Important**
>
> When creating a new connection, the default connection status is **Enabled** when you reach the **Activation & Summary** tab.
> Whether you choose to disable a new connection now or later, you must click **Save** on the **Activation & Summary** tab if you want to keep the new connection.

The **SSO Application Endpoint** provides a sample URL at the `/idp/startSSO.ping` application endpoint that webmasters or web application developers at your site can use to invoke single sign-on for the connection. For a list of supported parameters, see Viewing IdP application endpoints.

SP affiliations

A service provider (SP) affiliation is a SAML 2.0 specification that permits a group of service providers to make use of the same persistent name identifier for account linking.

SP affiliations are useful when multiple SPs share a business relationship in which users need services from each affiliated provider. By agreement among the affiliation members, the same pseudonym can be used to populate the `SAML_SUBJECT` of assertions sent to all of the SP partners contained in this affiliation.

> **Note**
>
> Each connection in the affiliation must be configured to use the same identity provider adapter instance for generating account links. For more information, see Managing authentication source mappings.

Related links

- Account linking

Managing SP affiliations

Use service provider (SP) affiliations when multiple SP’s share a business relationship in which users need services from each affiliated provider. To support these needs, edit and delete your SP affiliations in PingFederate as needed.

About this task

In **SP Affiliations** window, create, edit, or delete an SP affiliation.

Steps

- To create a new SP affiliation, go to **System > Protocol Metadata > SP Affiliations** to open the **SP Affiliations** window. Click **Create Affiliation**.

  Result:

  This is will open the **Create an Affiliation** window configuration.

- To edit or delete an SP affiliation, choose from the following options.

Choose from:

- To edit an SP affiliation, on the **SP Affiliations** window, select the affiliation by its ID and follow the configuration wizard to complete the task.
- To delete an SP affiliation, in the **SP Affiliations** window, click **Delete** under **Action** for the SP affiliation.

Importing affiliation metadata

You can import a metadata file for an identity provider (IdP) to send containing information that automatically specifies members of a service provider (SP) affiliation, describing this affiliation, to provide authentication for account linking.

About this task

On the **Import Metadata** tab, import a metadata file, or click **Next** if you do not want to import a file.

Steps

- Go to **System > Protocol Metadata > SP Affiliations** to open the **SP Affiliations** window. Click **Create Affiliation**.

  Result:

  This is will open the **Create an Affiliation** window configuration.

- To import a metadata file or not and move onto the next step, choose from the following options.

  Choose from:

  - To import a metadata file, on the **Import Metadata** tab, click **Choose File** to upload it. Click **Next**.
  - If you do not have a metadata file, on the **Import Metadata** tab, click **Next**.

Entering affiliation information

If you did not import a metadata file to describe a new service provider (SP) affiliation, enter the information manually to identify your SP affiliation to support authentication for account linking.

About this task

On the **Affiliation General Info** tab, enter the affiliation ID information, as described in the following table.

Steps

- To create a new SP affiliation, go to **System > Protocol Metadata > SP Affiliations** to open the **SP Affiliations** window. Click **Create Affiliation**.

  Result:

  This is will open the **Create an Affiliation** window configuration.

- On the **Affliation General Info** tab, enter the following information.

Affiliation ID fields and descriptions

Field	Description
Affiliation ID	A unique identifier for this affiliation. This value serves as the Name ID qualifier for SAML assertions sent to affiliated SP partners.
Affiliation Owner	Any SAML 2.0 SP connection can serve as the Owner.

If you imported a metadata file, this information is already supplied. However, you can change the Affiliation ID or select a different Affiliation Owner, if required.

When finished, click **Next**.

Managing affiliation membership

Manage the list of service provider (SP) connections that are part of this affiliation. Configure each of the SP connections in an affiliation to generate opaque pseudonyms using the same adapter attributes.

About this task

On the **Affiliation Membership** tab, create and manage a list of SP connections to be included in the affiliation.

If you imported a metadata file, this information is already supplied. However, you can add or remove connections from the affiliation.

Steps

- Go to **System > Protocol Metadata > SP Affiliations** to open the **SP Affiliations** window. Click **Create Affiliation**.

  Result:

  This is will open a new **Create an Affiliation** configuration window.

- Click on the **Affliation Membership** tab.
- To add an SP partner connection to the affiliation, from the **SP Connection Name** list, select the connection. Click **Add**.

> **Important**
>
> Each connection in the affiliation must be configured to use the same identity provider (IdP) adapter instance for generating account links. For more information, see Managing authentication source mappings.

- To remove a member of the affiliation, click **Delete** under **Action** for the connection.

> **Note**
>
> If you delete an affiliation member supplied by an imported metadata file and then save the affiliation, that connection will not appear in the drop-down list for re-adding in the future.

- When finished, click **Next**.

Reviewing an SP affiliation

Review the summary information for a service provider (SP) affiliation and make changes or save as needed.

About this task

On the **Activation & Summary**, review, amend, discard, or save your changes.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

OAuth configuration

OAuth is an open standard for authorization. To use PingFederate as an OAuth authorization server (AS), configure the OAuth AS settings as described in this section.

> **Tip**
>
> Service providers can also add OAuth capabilities to the Browser SSO configuration for identity provider (IdP) connection partners, see Configure OAuth attribute mapping.

Related links

- About OAuth

Configuring OAuth use cases

Administrators can configure PingFederate to support the OAuth grant types that applications require.

Steps

1. To configure the authorization server settings, go to **System** > **OAuth Settings** > **Authorization Server Settings**. For more information, see Configuring authorization server settings.
2. Define any number of optional common scopes and exclusive scopes, create scope groups from optional scopes as needed, and enter an optional description for the default scope in the **System** > **OAuth Settings** > **Scope Management** window.
3. Create one or more access token management instances in the **Applications** > **OAuth** > **Access Token Management** window.

   > **Note**
   >
   > You can also define the access token attribute contract for an access token management instance in this window.

4. Configure one or more entries to map attributes from authentication sources to the persistent grants.

Authorization Code or Implicit

- Map attributes from an identity provider (IdP) adapter instance to the persistent grants in **Authentication > OAuth > IdP Adapter Grant Mapping**.
- Map attributes from an IdP connection to the persistent grants in **IdP Connection > Browser SSO > OAuth Attribute Mapping**.
- Create an authentication policy contract (APC) using the **Policy Contracts** window, define an authentication policy to map attributes from the authentication sources (IdP adapter instances, IdP connections, or both) to the APC, and map attributes from the APC to the persistent grants using the **Authentication Policy Contract Grant Mapping** window.

> **Tip**
>
> If you are using a combination of authentication policies, APCs, and APC mappings, you can skip the **IdP Adapter Grant Mapping** and **OAuth Attribute Mapping** configurations.

Resource Owner Password Credentials

- Map attributes from a password credential validator instance to the persistent grants using the **Authentication > OAuth > Resource Owner Credentials Grant Mapping** configuration wizard.

> **Note**
>
> This is the first stage of the two-stage access token mapping process through the persistent grants.

5. Configure one or more entries to map attributes from the persistent grants (or the authentication sources directly) to the attribute contract of your access token management instances in the **Applications > OAuth > Access Token Mapping** window. Additionally, you can configure a mapping for clients using the client credential grant type.

> **Note**
>
> This is the second stage of the two-stage access token mapping process through the persistent grants. For more information about the access token mapping process, see Mapping OAuth attributes.

6. For the client-initiated backchannel authentication (CIBA) flow, configure one or more CIBA authenticator instances and then one or more CIBA request policies.

7. For the JSON web token (JWT) Bearer or SAML 2.0 Bear assertion grants flow, configure a mapping in **IdP Connection > OAuth Assertion Grant Attribute Mapping**.

> **Note**
>
> This use case exchanges a JWT or a SAML assertion for an OAuth access token.

8. Define one or more OpenID Connect policies using the **Applications > OAuth > OpenID Connect Policy Management** window if you support OpenID Connect use cases.

9. Go to **Applications > OAuth > Clients** and create one or more OAuth clients in the **Client** window.

10. **Optional:** Configure client settings and registration policies for dynamic client registration.

11. **Optional:** Configure client session management settings.

Configuring authorization server settings

The **Authorization Server Settings** page provides control over the usage and behavior of PingFederate as an authorization server, including the policies and settings for various grant types, refresh-tokens, persistent grants, and ID tokens.

Steps

1. Go to **System > OAuth Settings > Authorization Server Settings**.
2. Configure the authorization server to suit your use cases.

   The following table describes each field.

Field	Description
Authorization Code Timeout (Seconds)	The amount of time in seconds that an authorization code is considered valid. The default value is `60`.
Authorization Code Entropy (Bytes)	The length in bytes of the authorization code returned to clients. The default value is `30`.
Disallow the Plain PKCE Code Challenge Method (optional)	When selected, PingFederate, acting as the authorization server, rejects an authorization request that uses the PKCE `plain` `code_challenge_method`. By default, this checkbox is cleared. Learn more in Proof Key for Code Exchange by OAuth Public Clients.
Include Issuer in Authorization Response	When selected, PingFederate's issuer identifier `iss` parameter is returned in the authorization response. By default, this checkbox is cleared.
Track User Sessions for Logout	When selected, PingFederate links the sessions for identity provider (IdP) adapters that are used by clients to the PingFederate authentication session of the resource owner. When a user initiates logout, PingFederate sends logout requests to close the adapter sessions through the browser. Selecting this checkbox also lets you configure the logout mode and logout endpoints for each client. Learn more in **Logout Mode** in Configuring OAuth clients. By default, this checkbox is cleared.
Client Secret Retention Period	The amount of time in minutes that the previous client secret is retained after changing the secret. If value is `0`, client secrets will not be retained. Maximum value is one month or `43800` minutes. **Note** Global configuration can be overridden on a per client basis.
Pushed Authorization Request (PAR) Settings	

Field	Description
PAR Status	This field determines whether clients can or must use the PAR endpoint `/as/par.oauth2` on the authorization server to initiate authorization flows. The default setting is **Enabled**. This setting works in conjunction with each client's **Require Pushed Authorization Requests** checkbox on the **Client** page. For example: ◦ If PAR is **Enabled** on the authorization server and required on the client, then the client must use PAR. ◦ If PAR is **Enabled** on the authorization server but not required on the client, then the client can use PAR. ◦ If PAR is **Required** on the authorization server but not required on the client, then the client must use PAR. ◦ If PAR is **Disabled** on the authorization server and required on the client, then the client cannot access the authorization server. Learn more about PAR in Pushed authorization requests endpoint and Configuring OAuth clients.
PAR Reference Timeout (Seconds)	Specifies the lifetime of the request Uniform Resource Identifier (URI) that the PAR endpoint returns to the client. The default is 60 seconds. The allowable values are `1` to `1800`.
PAR Reference Entropy (Bytes)	Specifies the length of the request URI. Longer request URIs are more secure. The default is 24 bytes. The allowable values are `20` to `256`.
Refresh Token and Persistent Grant Settings	
Persistent Grant Max Lifetime	This field determines whether persistent grants should expire, and if so, the default maximum lifetime for persistent grants. ◦ Select **Grants Do Not Expire** to let persistent grants remain valid until they are revoked or removed. ◦ Select the time value option to set a maximum lifetime for persistent grants. Enter an integer between 1 and 999 in the field and select `Days`, `Hours`, or `Minutes` in the list. The default selection is **Grants Do Not Expire**. You can override the expiration in individual client records or grant-mapping configurations. Grant-mapping configurations take precedence and require an extended persistent grant attribute, `PERSISTENT_GRANT_LIFETIME`. ⓘ **Note** Grants expire at the max lifetime set when they are created. When a grant is reused, the max lifetime should be reused as well. New refresh token requests only update the `accessGrantUpdated` value, and won't change the grant's max lifetime.

Field	Description
Persistent Grant Idle Timeout	This field determines whether persistent grants should expire due to inactivity, regardless of whether the maximum lifetime has been reached, and the default idle timeout period. ◦ Select **Grants Do Not Timeout Due To Inactivity** and let persistent grants remain valid until they expire or are removed. ◦ Select the time value option to set the default idle timeout window. Enter an integer between 1 and 999 in the field and select a unit of measurement in the list. If you configure an idle timeout value, the idle timeout window slides when a persistent grant updates. When you have an idle timeout value configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity or until the grant storage revokes or removes them. When you have an idle timeout value configured with a maximum lifetime, persistent grants remain valid until they expire due to inactivity or lifetime expiration or until the grant storage removes them. Learn more in Transient grants and persistent grants. For new installations, the default inactivity allowance is 30 days. For upgrades, **Grants Do Not Timeout Due To Inactivity** is the default selection unless a specific value was set previously. ⓘ **Note** You can override the expiration in individual client records.
Refresh Token Length (Characters)	The length of the refresh tokens in numerical characters. The default value is `42`.
Roll Refresh Token Values (Default Policy)	When selected, PingFederate generates a new refresh token when a new access token is issued. Otherwise, each refresh token is used until it becomes invalid, either by manual revocation or another security setting that renders the token invalid. ⓘ **Note** New refresh tokens are not issued during the interval defined by the **Minimum Interval to Roll Refresh Tokens** field. By default, this checkbox is cleared.
Minimum Interval to Roll Refresh Tokens (Hours)	The minimum number of hours that must pass before a new refresh token can be issued. This setting provides a way to allow for rolling refresh tokens without having to send a new refresh token on every request. The default value is `0`.

Field	Description
Refresh Token Rolling Grace Period (seconds)	The amount of time in seconds that a rolled refresh token is still valid in the event that the client failed to receive an updated one during a roll. The maximum value is `86400`. For security reasons, you should set the custom value as low as possible. ⓘ **Note** If specified: ◦ The most recently issued refresh tokens must be refreshed once and converted to the new format. ◦ The global settings apply to all clients. The default value is 60.
Reuse Existing Persistent Access Grants for Grant Types	If a client makes multiple requests for the same user and the same or lesser scope, select the grant types that you want PingFederate to reuse the existing grant for, rather than creating a new grant for each request. Reusing an existing persistent grant imposes a limit of one grant per client, per user. In the context of refresh tokens, only the most recently issued is valid, and the previously issued refresh token is invalidated. If the same client are installed on multiple devices and used regularly by the same user, the grant type used by this client should be cleared. The applicable grant types are: ◦ **Implicit** (default) ◦ **Authorization Code** ◦ **Resource Owner Password Credentials** When the **Implicit** checkbox is selected, PingFederate requests consent from the user only once. The user is not asked for authorization on subsequent requests until the access grant is revoked. When the **Authorization Code** checkbox is selected, the same is true if the **Bypass Authorization for Previously Approved Persistent Grants** checkbox is also selected.
Allow Unidentified Clients to Make Resource Owner Password Credentials Grants	When selected, PingFederate allows resource owners to obtain access tokens without client ID or client authentication. By default, this checkbox is cleared.

Field	Description

Allow Unidentified Clients to Request Extension Grants	When selected, PingFederate allows user-initiated or client-initiated events, such as a mobile application or a scheduled task, to obtain access tokens without the client presenting a `client_id` or `client_secret` for the extension grant types, namely:
	◦ JSON Web Token (JWT) Bearer Token grant type
	`urn:ietf:params:oauth:grant-type:jwt-bearer`
	◦ SAML 2.0 Bearer Assertion grant type
	`urn:ietf:params:oauth:grant-type:saml2-bearer`
	◦ Validation grant type
	`urn:pingidentity.com:oauth2:grant_type:validate_bearer`
	By default, this checkbox is cleared.

Token Endpoint Base URL	When clients authenticate with the private_key_jwt authentication method, PingFederate validates the `aud` parameter value found inside the signed JWT against its base URL or any configured virtual host names. If the values do not match, the authentication fails. Enter a separate base URL that PingFederate can also take into consideration when validating the `aud` parameter. If configured, the OpenID Provider (OP) configuration endpoint `/.well-known/openid-configuration` uses the **Token Endpoint Base URL** field value as the base for the token endpoint. This field has no default value.
Persistent Grant Extended Attributes	

Field	Description
Attributes	Extend persistent grants to include additional attributes from your authentication systems. ***Lifetime of persistent grants*** Add the attribute `PERSISTENT_GRANT_LIFETIME` to enable the option to set the lifetime of persistent grants based on the outcome of attribute mapping expressions in individual grant-mapping configurations. For grant-mapping configurations that do not require this fine-grain control, you can configure them to use the default value.This capability applies to the following grant-mapping configurations: ◦ **IdP Adapter Grant Mapping** ◦ **OAuth Attribute Mapping** ◦ **Authentication Policy Contract Mapping** ◦ **Resource Owner Credentials Grant Mapping** This field has no default entry. PingFederate allows multiple entries. ⓘ **Note** If you have already created grant mapping configurations and then add one or more attributes in this section, the newly added attributes are configured as **No Mapping** in all existing grant mapping configurations. You can configure fulfillment for the newly added attributes in individual grant mapping configurations when your use cases require those attributes.
Authorization Consent	
Bypass Authorization for Previously Approved Persistent Grants	When selected, PingFederate requests consent from the user only once. The user is not asked for authorization on subsequent requests until the access grant is revoked. This applies only when using the authorization code grant type and when the **Reuse Existing Persistent Access Grants for Grant Types** checkbox is selected. By default, this checkbox is cleared.

Field	Description
Consent User Interface	Specifies whether PingFederate or a trusted web application should handle consent approval. ***Default*** Select **Default** and let PingFederate handle consent approval by presenting the **Request for Approval** page to the resource owner. ***External*** Select **External** to delegate the responsibilities of consent approval to a trusted web application. For example, if you have created an instance of the Reference ID Adapter version (1.5 or later), you can select it in the list. The expectation is that the trusted web application is integrated with PingFederate through this Reference ID Adapter instance. When selected, you must also configure two additional fields: **External Consent IdP Adapter** and **External Consent Scopes Attribute**. ***External Consent IdP Adapter*** The **External Consent IdP Adapter** field displays a list of IdP adapter instances that are capable of facilitating the consent approval process. For example, if you have created an instance of the Reference ID Adapter version (1.5 or later), you can select it in the list. The expectation is that the trusted web application is integrated with PingFederate through this Reference ID Adapter instance. Your development team can also create a custom adapter using the PingFederate SDK. Learn more in the Javadoc about the `IdpAuthenticationAdapterV2` interface, the `ExternalConsentPageAdapter.java` file for a sample implementation, and the SDK Developer's Guide for build and deployment information. **Tip** The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory. After you've deployed, you can create an instance of the selected adapter instance. ***External Consent Scopes Attribute*** The **External Consent Scopes Attribute** field displays a list of attributes defined in the IdP adapter contract of the selected adapter instance. Select the attribute whose value contains the approved scopes returned by the trusted web application. For example, if you have added an attribute called **approvedScopes** to the adapter instance with the expectation that **approvedScopes** is the attribute that the trusted web application passes approved scopes to PingFederate, select **approvedScopes** in the list.
OAuth Administrative Web Services Settings	

Field	Description
Password Credential Validator	Selects a password credential validator (PCV) instance to manage clients using the OAuth Client Management Service or to manage persistent grants using the OAuth Access Grant Management Service. This setting has no default selection. When no PCV is selected, neither service can be used.
Persistent Grant Management API The Persistent Grant Management API allows clients to assume the responsibility of grant management if the users authorize the clients to do so. In this scenario, a client prompts the user to approve a specific scope for managing persistent grants on the user's behalf. If the user approves, the client requests an access token with such scope from PingFederate. As long as the access token remains valid, the client can retrieve and revoke persistent grants and their associated extended attribute names and values for that user. Learn more in OAuth Persistent Grant Management API.	
Access Token Manager	Selects an Access Token Management instance under which one or more clients can use the access tokens issued to manage persistent grants on their users. Such clients must also be configured to use this Access Token Management instance in their client configurations.
Required Scope	Selects the scope that PingFederate looks for in access tokens prior to granting clients the permission to manage persistent grants for their users. Clients must obtain access tokens with this scope and include them in their grant-management requests.
Cross-Origin Resource Sharing Settings	

Field	Description
Allowed Origin	Enter any number of trusted origins to enable cross-origin resource sharing (CORS) support for the following OAuth endpoints: ◦ `/as/token.oauth2` ◦ `/as/revoke_token.oauth2` ◦ `/idp/userinfo.openid` ◦ `/pf-ws/rest/oauth/grants/` ◦ `/pf/JWKS` ◦ `/.well-known/openid-configuration` ◦ `/as/bc-auth.ciba` After configuration, client-side web applications from the trusted origins are allowed to make requests to the PingFederate authorization server for the purpose of accessing protected resources, such as obtaining or renewing access tokens with refresh tokens, presenting access tokens for revocation, querying additional claims (user attributes), and retrieving OpenID Provider configuration information and JavaScript Object Notation (JSON) Web Key Sets. Learn more about CORS in W3C's recommendation on Cross-Origin Resource Sharing. Here are some example entries and their behaviors: ◦ `https://www.example.com` CORS requests originating from `https://www.example.com` are allowed. ** `https://www.example.com:8080` CORS requests originating from `https://www.example.com:8080` are allowed. * `https://www.example.com:` CORS requests originating from `https://www.example.com:<any port>` are allowed. However, a port number is required in the `Origin` request header. **Important** Although using the wildcard character provides the convenience of allowing multiple origins with one entry, consider adding individual origins to limit CORS requests to a list of trusted hosts. This field has no default entry. PingFederate allows multiple entries.
Device Authorization Grant Settings The OAuth 2.0 Device Authorization Grant specification defines the process of a user granting authorization to a device using a browser on a second device, such as a smartphone or a computer. Learn more in Device authorization grant.	

Field	Description
User Authorization URL (Optional)	This field determines whether PingFederate should use a different URL, possibly for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses. Learn more in Device authorization endpoint. For example, if this field is configured with a value of `https://www.example.org/welcome`, PingFederate returns https://www.example.org/welcome and https://www.example.org/welcome?user_code=<*activationcode*> as the verification URIs. After processing the device authorization response, which includes the verification URIs, the device presents one of them to the user. The user is expected to browse to the presented verification URI on a second device. **Important** The target web server must redirect the browser to PingFederate at its user authorization endpoint. Learn more in User authorization endpoint. The target web server must also preserve the `user_code` parameter value, if provided. For instance, if the base URL of your PingFederate server is https://www.example.com and this field is configured with a value of `https://www.example.org/welcome`, the target web server must redirect as follows: ◦ https://www.example.org/welcome to https://www.example.com/as/user_authz.oauth2 ◦ https://www.example.org/welcome?user_code=<*activationcode*> to https://www.example.com/as/user_authz.oauth2?user_code=<*activationcode*> This field has no default value. **Note** You can override this setting in individual client records.
Registered Authorization Path (Optional)	This field controls whether PingFederate should replace the user authorization endpoint with a different path, perhaps for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses. Learn more in Device authorization endpoint. The domain portion remains to be the base URL of PingFederate. For example, if the base URL is https://www.example.com and this field is configured with a value of `/go`, PingFederate returns https://www.example.com/go and https://www.example.com/go?user_code=<*activationcode*> as the user authorization URLs. If PingFederate receives a device authorization request at one of the configured virtual host names, PingFederate preserves the virtual host name in its device authorization responses. **Note** The registered authorization path behaves the same way as the user authorization endpoint does. Learn more in User authorization endpoint. This field is ignored when the **User Authorization URL** field is configured here or in the individual client records. The configured value must begin with a forward slash. This field has no default value.

Field	Description
Pending Authorization Timeout (seconds)	The lifetime of an activation code, the `user_code` parameter value, in seconds. The default value is `600`. **Note** You can override this setting in individual client records.
Device Polling Interval (seconds)	The amount of time in seconds that the device waits between polling requests to the PingFederate token endpoint. The default value is `5`. **Note** You can override this setting in individual client records.
Check Activation Code	This setting determines whether the activation code verification step should happen before or after the user is authenticated. One advantage of checking before authentication is that it allows PingFederate to determine up front the client ID and scopes for the flow. As a result, selectors such as the OAuth Client Set Authentication Selector, Extended Property Authentication Selector, and OAuth Scope Authentication Selector will work as expected in deciding the authentication policy for the user. The default setting is **After Authentication**. **Note** Checking the activation code before authentication means that the key PingFederate uses for rate limiting is based only on the user's IP address. This means that the rate limit must be higher, because the key will be shared across all users who have the same IP address. Learn more through the `<pf_install>/pingfederate/server/default/data/config-store/oauth-device-flow.xml` file, including how to tune the rate limit appropriately for your deployment.
Bypass Activation Code Confirmation	When PingFederate receives a verification request that includes an activation code, the `user_code` parameter value, it prompts the user to confirm the activation code. This field determines whether PingFederate skips this confirmation step. Select the **Bypass Activation Code Confirmation** checkbox for PingFederate to skip the confirmation step. By default, this checkbox is cleared. **Note** You can override this setting in individual client records.
JWT Secured Authorization Response Mode (JARM) JARM is a mechanism to enhance the security of the standard authorization response. It adds support for signing and encryption, sender authentication, and audience restriction. It also offers protection from replay, credential leakage, and mix-up attacks. JARM can be combined with any response type. Learn more through the JARM specification.	
JWT Lifetime (Seconds)	The lifetime of the authorization response token that PingFederate returns. The default value is `600` seconds.

Field	Description
Demonstrating Proof-of-Possession (DPoP) The DPoP protocol provides a way for clients to prove possession of a private key associated with a specific access token during each request. By including a DPoP token in the request, the authorization server can verify that the client has legitimate access to the private key. You can require a client to use DPoP by selecting the **Require DPoP** checkbox on the **Client** page. Learn more about the protocol through the OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer specification.	
Require DPoP Proof JWT Nonce	Specifies whether a nonce is required in the DPoP proof JWT. By default, the nonce is not required.
DPoP Proof JWT Lifetime (Seconds)	Specifies the lifetime of the DPoP proof JWT. By default, the lifetime is `120` seconds.
Enforce DPoP Proof JWT Replay Prevention	Specifies whether PingFederate requires a unique signed DPoP proof JWT from the client for each request. By default, the checkbox is cleared. **Note** The underlying Assertion Replay Prevention Service is cluster-aware. Learn more in Assertion Replay Prevention Service.

External consent user interface

As use cases evolve and give users more control over their data, it is important to provide detailed information about the requests. In addition to scope and authorization detail descriptions, PingFederate supports the use of an external web application to prompt for authorization consent.

An external web application provides the opportunity to retrieve additional information specific to the users. For example, if a client requests the **read_bank_account** scope, the web application can retrieve the user’s customer information file and give the user the ability to choose which accounts to be made available to the client.

Authorization details are used in a similar way as scopes. For information about authorization details, see OAuth rich authorization requests.

To use an external web application for consent approval, configure the **Consent User Interface** setting:

1. Go to **System > OAuth Settings > Authorization Server Settings**.
2. For the **Consent User Interface** setting, select **External**.
3. Select an **External Consent IdP Adapter**.
4. Select an **External Consent Scopes Attribute**, or an **External Consent Authorization Details Attribute**, or one of each.

Responsibilities of the external web application

Delegating consent approval to an external web application implies that PingFederate can trust the web application. PingFederate expects this trusted web application to fulfill the following responsibilities:

- Retrieve from PingFederate the list of requested scopes and authorization details in a secure manner.

  For example, when integrating the web application with PingFederate through an instance of the Reference ID Adapter, such communications occur through a direct connection between the web application and PingFederate. This back-channel connection is protected by authentication and encryption (HTTPS).

- Provide to the resource owner the information associated with the list of requested scopes and authorization details, and the user interface elements to approve or deny them.
- Validate that the approved scopes and authorization details found in the response from the resource owner do not exceed the requested scopes and authorization details.

  > **Important**
  >
  > This validation guards against unauthorized access in the event that the response is tampered with and the original approved scopes and authorization details are compromised.

- As needed, modify the approved scopes and authorization details before returning them to PingFederate.

  This allows the web application to override authorization decisions.

- Return the list of approved scopes and authorization details to PingFederate in a secure manner.

Handling of approved scopes and authorization details

By default, PingFederate handles consent approval by presenting the **Request for Approval** page to the resource owner. Upon receipt of the response from the resource owner, PingFederate validates that the approved scopes and authorization details do not exceed those requested. If the validation passes, PingFederate adds the approved scopes and authorization details to the access token. Otherwise, PingFederate returns an `invalid_scope` error or `invalid_authorization_details` error.

When an external consent user interface is enabled, PingFederate delegates consent approval to an external web application. As PingFederate trusts this web application, it always adds the scopes and authorization details returned by the trusted web application to the access token, regardless of whether the returned scopes and authorization details have already been defined in the system. The issuance of the access token is still subject to the criteria defined in the grant mapping configuration, the token mapping configuration, or both. For more information, see Grant contract mapping and Token mapping.

Scopes and scope management

OAuth allows you to constrain the privileges associated with an access token, and scopes allow you to define the privileges requested and granted.

Static scopes versus dynamic scopes

As an authorization server, PingFederate supports the concepts of static scopes and dynamic scopes. To define a static scope, use a text value such as `read_bank_account`. To define a dynamic scope, use a text value with a variable component represented by a wildcard, such as `read_bank_account_txn:*`. As illustrated, dynamic scopes allow clients to request authorization using scope values with a variable component from one request to another.

For example, when ah a requested scope of `read_bank_account_txn:1234`, PingFederate can match the requested scope to the dynamic scope pattern of `read_bank_account_txn:*` and can issue an access token with the requested scope of `read_bank_account_txn:1234`.

Scope groups

For ease of management and subsequent client interactions, PingFederate has the capability to create multiple groups of static scopes. A client can reference a scope group in applicable OAuth 2.0 protocol interactions. When authorized, clients can subsequently request access tokens with fewer permissions by presenting to the token endpoint a refresh token and the desired subset of scopes.

A scope group must contain at least one static scope, and multiple sub scopes are allowed. Multiple scope groups can share the same set of sub scopes. However, no scope group can contain another scope group or the default scope.

Scope group expansion

An authorization request can include one or more scope values. If the request is authorized, PingFederate issues an access token to the client. When the client brings the access token to a resource server to access protected resources, the resource server may contact PingFederate to validate the access tokens. Scope groups are not expanded in JSON web token (JWT)-based access tokens or token introspection responses by default. You can optionally enable scope group expansion per access token management instance.

> **Note**
>
> Regardless of whether you choose to expand scope groups, the **Request for Approval** window always presents the description of the requested scope groups, if any.

Common scopes and exclusive scopes

PingFederate has the flexibility to manage common and exclusive scopes and scope groups.

Common scopes and scope groups

Common scopes and scope groups are optional. If defined, they are available to all clients by default. As needed, you can restrict individual clients to a subset of common scopes or scope groups in their configurations.Clients created via the Dynamic Client Registration protocol can also be restricted to a subset of common scopes or scope groups based on the configuration on the **Scope Constraints** tab in **System** > **OAuth Settings** > **Client Settings**. The **Scope Constraints** configuration is shared across all clients registered through dynamic client registration. If a certain client requires a different set of common scopes or scope groups, modify the client configuration by using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created.

Exclusive scopes and scope groups

Exclusive scopes and scope groups are optional. If defined, they are restricted from all clients by default. As needed, you can configure individual clients to allow a subset of exclusive scopes or scope groups in their configurations.Clients created with the Dynamic Client Registration protocol can also be configured to allow a subset of exclusive scopes or scope groups based on the **Scope Constraints** tab on the **System** > **OAuth Settings** > **Client Settings** window. The **Scope Constraints** configuration is shared across all clients registered via dynamic client registration. If a certain client requires a different set of exclusive scopes or scope groups, modify the client configuration by using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created.

Note

A scope or scope group is either a common scope or group, or an exclusive scope or group. Duplicate scopes and scope groups are not allowed. Scope and scope group values are case-sensitive.

Tip

Create scopes that are intended for the majority of clients as common scopes. Create scopes that should be limited to the minority of clients as exclusive scopes.

OpenID Connect

If one or more clients support the OpenID Connect standard, add the following scopes for the purpose of requesting specific sets of claims from the OpenID Provider:

- `openid`
- `address`
- `email`
- `phone`
- `profile`

Tip

If most clients are allowed to use these scopes, create them as common scopes.

Per-client scope management

You can manage scope access on a client-to-client basis. The client settings are **Restrict Common Scopes** and **Exclusive Scopes**.

Restrict Common Scopes

This setting determines whether all common scopes and scope groups should be made available to the client, or only a select few.When selected, the administrative console displays a list of existing common scopes and scope groups. Choose the common scopes and scope groups that are intended for the client. The rest, as well as any future common scopes and scope groups, become invalid for the client. If the client tries to use such scope or scope group, it will receive an `invalid_scope` error message from PingFederate.When cleared, all existing and future common scopes and scope groups are available to the client. This is the default behavior.

Exclusive Scopes

This setting determines whether any exclusive scopes and exclusive scope groups should be made available to the client.When selected, the administrative console displays a list of existing exclusive scopes and scope groups. Choose the exclusive scopes and scope groups that are intended for the client. The rest, as well as any future exclusive scopes and scope groups, become invalid for the client. If the client tries to use such scope or scope group, it will receive an `invalid_scope` error message from PingFederate.When cleared, no exclusive scopes and scope groups are available to the client. This is the default behavior.

Note

Both settings impact dynamic scope evaluation. For more information, see Dynamic scope evaluation and per-client scope management.

Dynamic scopes

A dynamic scope is defined by using a text value with a variable component represented by an asterisk (`*`). PingFederate supports three dynamic scope patterns:

- A prefix followed by a wildcard, for example: `prefixTextValue*`
- A wildcard followed by a suffix, for example: `*suffixTextValue`
- A wildcard placed between a prefix and a suffix, for example: `prefixTextValue*suffixTextValue`

PingFederate only allows one variable component. Backslashes (`\`) and double quotation marks (`"`) are not allowed in the prefix or the suffix.Multiple dynamic scopes are supported in conjunction with any number of static scopes and scope groups.

Dynamic scope evaluation

When a client sends an authorization or token request with a list of desired scopes, PingFederate validates the requested scopes against its configurations.

If PingFederate finds no match for the requested scopes, it returns an `invalid_scope` error message to the client.

If PingFederate matches the requested scope to an existing static scope or scope group, it checks the client configuration to determine whether such static scope or scope group is valid for the client. If it is, PingFederate proceeds further. For example, if PingFederate is configured to handle consent approval, it presents to the user the **Request for Approval** window with the description associated with the matched static scope or scope group. If PingFederate should issue an access token, the token is issued with the requested scope. If such static scope or scope group is not valid for the client, PingFederate returns an `invalid_scope` error message to the client.

If PingFederate finds no exact match but finds a partial match to one or more dynamic scopes, the partial match with the highest number of matched characters in the prefix, suffix, or both is the matched dynamic scope. In the event that two partial matches tie, the partial match with the highest number of characters matched in the prefix is the matched dynamic scope. PingFederate then checks the client configuration to determine whether such dynamic scope is valid for the client. If it is, PingFederate proceeds further. Otherwise, PingFederate returns an `invalid_scope` error message to the client. If PingFederate should issue an access token, the token is issued with the requested scope, not the matched dynamic scope pattern.

Example

For example, you add the following dynamic scopes:

Common Scopes	Exclusive Scopes
`*123`	`zSomeExclusiveScope`
`*12345`	
`a*c#123`	
`ab*#123`	

Common Scopes	Exclusive Scopes
xy*123	
xy*	

You also add a client without any common scope restrictions. This client can access all common scopes.

The following table illustrates the expected results when the client sends an authorization request with these scopes:

- xy#1
- xy#12
- xy#123
- xy#1234
- xy#12345
- xy#123456
- xyz
- z123
- z12345
- abc#123

Requested scope	Matched dynamic scope	Variable component from the requested scope
xy#1	xy*	#1
xy#12	xy*	#12
xy#123	xy*123	#
xy#1234	xy*	#1234
xy#12345	*12345	xy#
xy#123456	xy*	#123456
xyz	xy*	z
z123	*123	z
z12345	*12345	z
abc#123	ab*#123	b

The minimum length of the variable component is one character. If the variable component contains two or more characters, it may also contain the asterisk character as well. Given the same common dynamic scopes and the same client configuration, requested scopes of `xyQ123`, `xy*Q123`, `xyQ*123`, `xy**Q*123` will be matched as in the following table.

Requested scope	Matched dynamic scope	Variable component from the requested scope
`xyQ123`	`xy*123`	`Q`
`xy*Q123`	`xy*123`	`*Q`
`xyQ*123`	`xy*123`	`Q*`
`xy**Q*123`	`xy*123`	`*Q`

If the client sends an authorization request with a requested scope of `xy*123`, it will receive an `invalid_scope` error from PingFederate.

Dynamic scope evaluation and per-client scope management

Depending on the configured dynamic scope patterns and if they are defined as common or exclusive dynamic scopes, per-client scope management settings can impact the results of scope evaluation.

The **Restrict Common Scopes** setting determines whether all common scopes and scope groups are available to the client, or only a select few. Use this setting to restrict certain common dynamic scopes.

The **Exclusive Scopes** setting determines whether any exclusive scopes and scope groups are available to the client. When this check box is not selected, PingFederate does not consider any exclusive dynamic scopes or any exclusive static scopes and scope groups when trying to match a requested scope against a list of configured scopes and scope groups. When the check box is selected, all exclusive scopes and scope groups are considered. If PingFederate matches a requested scope to an exclusive dynamic scope and such scope is not available to the client, PingFederate returns an `invalid_scope` error message to the client. This remains true for a lesser partial match to an available common dynamic scope.

Example

For example, you update your previous sample scope configuration as in the following table.

Common Scopes	Exclusive Scopes
`*123`	`xy*123`
`*12345`	`zSomeExclusiveScope`
`a*c#123`	
`ab*#123`	
`xy*`	

The following table describes the results when the client sends an authorization request with a requested scope of `xy#123`.

Per-client scope management settings	Result	Variable component from the requested scope
Restrict Common Scopes - Not selected. **Exclusive Scopes** - Not selected.	PingFederate matches the requested scope of `xy#123` to the common dynamic scope of `*123`. The exclusive dynamic scope of `xy*123` is not taken into consideration because the **Exclusive Scopes** check box is not selected. PingFederate does not consider any exclusive scopes and scope groups as eligible candidates in this scenario.	`xy#`
Restrict Common Scopes - Not selected. **Exclusive Scopes** - Selected. ◦ `zSomeExclusiveScope` is selected. No other exclusive scope is selected.	PingFederate returns an `invalid_scope` error message, because the exclusive dynamic scope of `xy*123` is not allowed based on the **Exclusive Scopes** configuration.	Not applicable.
Restrict Common Scopes - Not selected. **Exclusive Scopes** - Selected. ◦ `xy*123` is selected. It does not matter if any other exclusive scope is selected.	PingFederate matches the requested scope of `xy#123` to the exclusive dynamic scope of `xy*123`.	`#`
Restrict Common Scopes - Selected. ◦ `xy*` is selected No other common scope is selected. **Exclusive Scopes** - Not selected.	PingFederate returns an `invalid_scope` error message, because the common dynamic scope of `*123` is not allowed based on the **Restrict Common Scopes** configuration. The exclusive dynamic scope of `xy*123` is not taken into consideration because the **Exclusive Scopes** check box is not selected. PingFederate does not consider any exclusive scopes and scope groups as eligible candidates in this scenario.	Not applicable.

Per-client scope management settings	Result	Variable component from the requested scope
Restrict Common Scopes • Selected. ◦ `*123` is selected It does not matter if any other common scope is selected. **Exclusive Scopes** • Selected. ◦ `xy*123` is selected. It does not matter if any other exclusive scope is selected.	PingFederate matches the requested scope of `xy#123` to the exclusive dynamic scope of `xy*123`, because `xy*123` is a better partial match than `*123`.	`#`

Description for scopes and scope groups

When defining a scope or a scope group, enter a value and a description for the scope or the scope group. This description helps you identify the purpose of the scope or scope group at a later time. If PingFederate is configured to handle consent approval, the **Scope Description**, **Scope Group Description**, and **Default Scope Description** fields determines the text that appears on the **Request for Approval** window.

Default scope

The default scope is the implied permissions when no scope and scope group values are indicated, or in addition to any scope or scope group values.If your organization requires a localized description, enter a unique alias in the **Default Scope Description** field, such as `oauth.approval.page.template.defaultScope`. Insert the same alias with the desired localized text in the applicable language resource files, located in `<pf_install>/pingfederate/server/default/conf/language-packs`.

Static scopes and scope groups

You can enter simple descriptions or localize the descriptions by using the PingFederate localization framework.

Dynamic scopes

You can enter simple descriptions. You can also use a mix of text and scope-description variables. `${scope}` represents the requested scope, and `${scope-var}` represents the variable component found in the requested scope.Suppose you added a dynamic scope with a pattern of `dynaGet67*10` and a scope description of `${scope} contains ${scope-var}`. If a client requests a scope value of `dynaGet67eight910`, the resulting scope description is `dynaGet67eight910 contains eight9`. (`eight9` is the variable component found in the requested scope.)If your organization requires a localized description, enter a unique alias in the **Scope Description** field (for example, `oauth.approval.page.template.someDynamicScope`). Then insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory. You may also use scope-description variables as part of the localized text.

> **Note**
>
> Both scope-description variables are intended for dynamic scopes only. When they are used as description for static scopes or a scope groups, PingFederate shows them on the **Request for Approval** window.

Dynamic scopes and consent user interface

The default consent approval process and user interface in PingFederate are capable of handling dynamic scopes and their scope descriptions. While the scope description and the optional scope-description variables provide the basic controls to describe a given scope, PingFederate also supports the use of an external web application to prompt for authorization consent. This allows you to retrieve additional information specific to the users and apply application-specific scope-processing logic.

Coordinating with developers

Regardless of whether a static scope, a scope group, or a dynamic scope is created as common or exclusive, a scope or a scope group represents access to a resource or API on the RS. Applicable scope or scope group values require coordination with developers that are familiar with the details of the RS OAuth implementation. For clients supporting the OpenID Connect protocol, you can direct the developers to your PingFederate OpenID Provider configuration endpoint to retrieve a list of common scopes and common scope groups.

> **Note**
>
> The OpenID Provider configuration endpoint does not return exclusive static scopes, exclusive scope groups, common dynamic scopes, and exclusive dynamic scopes by default. You can optionally customize the response to include such scopes and scope groups as needed.

Related links

- Configuring dynamic client registration settings
- Configuring scope constraints
- Configuring OAuth clients
- Localizing messages for end users
- External consent user interface
- OpenID Connect, Requesting Claims using Scope Values (openid.net/specs/openid-connect-core-1_0.html#ScopeClaims)
- OpenID Provider configuration endpoint
- Customizing a configuration endpoint response
- Introspection endpoint

Defining scopes

PingFederate manages scopes and scope groups in common and exclusive buckets.

About this task

Common scopes and scope groups are optional. They are available to all clients by default. You can restrict individual clients to a subset of common scopes or scope groups on a client-by-client basis in their client configurations.

Exclusive scopes and scope groups are also optional. They are restricted from all clients by default. However, you can grant individual clients access to one or more exclusive scopes or scope groups in their client configurations.

You can also create static scopes, static scope groups, and dynamic scopes. Scope groups allow clients to request a "super scope" and optionally downgrade to a subset of it later. Dynamic scopes address the business requirement where clients want to request authorization by using scope values with a variable component from one request to another. For detailed information about scopes, see Scopes and scope management.

You can manage scopes, scope groups, and the default scope description from **System > OAuth Settings** on the **Scope Management** window. Configuration steps for common and exclusive scopes and scope groups are identical.

> **Note**
>
> A scope or scope group is either a common scope or group, or an exclusive scope or group. Duplicate scopes and scope groups are not allowed.

> **Tip**
>
> Create scopes that are for the majority of clients as common scopes. Create scopes that are for a minority of clients as exclusive scopes. You can organize common or exclusive static scopes into common or exclusive scope groups.

Steps

1. Go to **System > OAuth Settings > Scope Management**.
2. On the **Common Scopes** or **Exclusive Scopes** tab, configure any number of common or exclusive static scopes.
    1. Enter a static scope value and a description in the **Scope Value** and **Scope Description** fields.

        Scope values are case-sensitive. A requested scope value of `email` does not match a configured static scope value of `Email`. Do not use backslashes (`\`) or double quotation marks (`"`) in the **Scope Value** field.

        > **Note**
        >
        > If PingFederate is configured to handle consent approval and your organization requires a localized description, enter a unique alias in the **Scope Description** field. For more information, see Description for scopes and scope groups.
    2. Click **Add**.
    3. Repeat to configure additional static scopes. Display order does not matter.
3. On the **Common Scopes** or **Exclusive Scopes** tab, configure any number of common or exclusive dynamic scopes.
    1. Enter a dynamic scope pattern and a description in the **Scope Value** and **Scope Description** fields.

        You must use a case-sensitive text value with the asterisk character (`*`). For supported dynamic scope patterns, see Dynamic Scopes.

        You can enter a simple description, or you can use a mix of text and scope-description variables. For more information, see Description for scopes and scope groups.

> **Note**
>
> If your organization requires additional information specific to the users or application-specific scope-processing logic, you can configure PingFederate to use an external web application to handle consent approval.

    2. Select the check box under **Dynamic**.

    3. Click **Add**.

    4. Repeat to define additional dynamic scopes. Display order does not matter.

4. On the **Common Scopes** or **Exclusive Scopes** tab, configure any number of common or exclusive scope groups.

    1. Enter a scope group value and its description under **Scope Group Value** and **Scope Group Description**.

       If PingFederate is configured to handle consent approval and your organization requires a localized description, enter a unique alias in the **Scope Group Description** field. For more information, see Description for scopes and scope groups.

       > **Note**
       >
       > All scope groups are defined as static scope groups. The partial-matching concept is intended for dynamic scopes only and does not apply to scope groups. Like static scope values, scope group values are case-sensitive. Do not use backslashes (`\`) or double quotation marks (`"`) in the **Scope Group Value** field.

    2. Select at least one static scope under **Sub Scopes**.

       > **Note**
       >
       > The administrative console filters out dynamic scopes because the scope-grouping capability is reserved for static scopes only.

    3. Click **Add**.

    4. Repeat to define additional scope groups. Display order does not matter.

5. **Optional:** On the **Common Scopes** or **Exclusive Scopes** tab, use the **Edit**, **Update**, and **Cancel** buttons to make or undo a change to an existing entry. Use the **Delete** and **Undelete** buttons to remove an existing entry or cancel the removal request.

   > **Caution**
   >
   > Updating or removing a scope or scope group value in production can cause runtime errors when a client request specifies the scope or scope group using the previously defined value. Errors can also occur when the requesting client is restricted to a scope or scope group that no longer exists unless the affected client configuration is also updated.

6. On the **Default Scope** tab, enter a description for the default scope.

   If PingFederate is configured to handle consent approval and your organization requires a localized description, enter a unique alias in the **Default Scope Description** field. For more information, see Description for scopes and scope groups.

> **Note**
>
> The default scope is the implied permissions when no scope and scope group values are indicated, or in addition to any scope or scope group values.

7. Click **Save**.

Result

Scopes and scope groups represent access to resources or APIs on the resource server . For clients supporting the OpenID Connect protocol, you can direct the developers to your PingFederate OpenID Provider configuration endpoint to retrieve a list of common scopes and common scope groups.

> **Note**
>
> By default, the OpenID Provider configuration endpoint does not return exclusive static scopes, exclusive scope groups, common dynamic scopes, and exclusive dynamic scopes. You can customize the response to include such individual scopes and scope groups.

Related links

- Configuring OAuth clients
- Localizing messages for end users
- OpenID Provider configuration endpoint
- Customizing a configuration endpoint response
- External consent user interface

Adding virtual issuers for OpenID Connect

You can define one or more virtual issuers for OpenID Connect, with or without a relative path. When minting an ID token, PingFederate populates the issuer claim according to the virtual issuer setting and the authorization request.

About this task

To add a virtual issuer to PingFederate, perform the following procedure. If you have multiple virtual issuers, ensure the combination of host and path values are unique.

> **Note**
>
> After you define virtual issuers, you can map them to sets of ID token signing keys. For more information, see Mapping ID token signing keys to virtual issuers.

Steps

1. Go to **System** > **OAuth Settings** > **Virtual Issuers**.
2. Click **Add Virtual Issuer**.
3. Enter a unique issuer **Name**.

4. Enter the **Host**.

5. Optional: Enter the relative **Path**, which must start with the value of the `pf.runtime.context.path` property in the `run.properties` file.

6. Click **Save**.

Configuring client settings

Use the **Client Settings** window to configure dynamic client registration settings.

Steps

1. On the **Dynamic Client Registration** tab, modify settings as needed.

2. Click **Next** to advance to the next tab.

3. Click **Save** to retain your changes.

Configuring dynamic client registration settings

Dynamic client registration allows developers to register OAuth clients through an API based on open standards.

About this task

PingFederate supports various client metadata as described in Supported client metadata. If specific use cases require additional metadata, add them as extended properties in **System > Server > Extended Properties**.

> **Important**
>
> As dynamic client registration can expose your server to unwanted client registrations, we recommend protecting PingFederate by requiring an initial access token, configuring one or more client registration policies, and protecting access to the dynamic client registration endpoint.

> **Note**
>
> Dynamic client registration requires OAuth client storage in an external datastore, such as a database or LDAP directory. If you have not yet switched from the default on-disk client storage to an external datastore, see OAuth client datastores. You can continue with the rest of the configuration; however, dynamic client registration remains inactive until an external client storage is defined.

Steps

1. Go to **System > OAuth Settings > Client Settings** and click **Dynamic Client Registration**.

2. Select the check boxes for the options that you want to use.

   The following table describes each option.

Option	Description
Enable Dynamic Client Registration	Select this option to enable dynamic client registration. This option is disabled by default.
Require Initial Access Token	Select this option to require an initial access token. If selected, you must also select the required scope or scope group from the list. **Important** Although optional, selecting this option can prevent unwanted client registrations. **Note** Developers must be set up to obtain access tokens with the required scope or scope group from your PingFederate authorization server. For example, you can create a new OAuth client for a group of developers, assign this client a specific scope for the purpose of creating other clients using the OAuth 2.0 Dynamic Client Registration protocol, and let the developers obtain their access tokens directly by completing one of the supported OAuth flows. You can also write a custom web application that uses the OAuth flow to obtain access tokens on behalf of the developers as they make their requests. This option is enabled by default.
Enable Dynamic Client Registration Management	Select this option to enable dynamic client registration management, and to make the following three client management/maintenance options visible. ◦ **Rotate Client Secret** - Select this option to rotate the client secret when a client updates or retrieves its configuration. This option is enabled by default. ◦ **Rotate Registration Access Token** - Select this option to rotate the registration access token when a client updates or retrieves its configuration. This option is enabled by default. ◦ **Allow Client Delete** - Select this option to allow clients to deprovision themselves on the authorizations server. This option is enabled by default. ◦ **Retain Client Secret** - Select this option to allow the client secret to be temporarily retained after a change. This option is not enabled by default. When enabled, **Client Secret Retention period** can be configured on the **Client Configuration Defaults** page. Dynamic client registration management allows a client to retrieve its configuration through `GET` requests, to update its configuration through PUT requests to the provided registration client URI, and to deprovision itself through a `DELETE`. For more information, see the OAuth 2.0 Dynamic Client Registration Management Protocol. This option is disabled by default.

Result

When dynamic client registration is active, developers can send client registrations to the `/as/clients.oauth2` endpoint to create OAuth clients dynamically.

Other maintenance calls can be made to the `registration_client_uri` returned in the original registration response. This endpoint has the format `/as/clients.oauth2/<clientId>` .

Related links

- OAuth 2.0 Dynamic Client Registration Protocol (tools.ietf.org/html/rfc7591)
- OpenID Connect Dynamic Client Registration, Client Metadata (openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata)
- Dynamic client registration endpoint
- Defining scopes
- Extended properties

Supported client metadata

PingFederate supports various client metadata, as described in the following table.

Metadata field	Metadata description
client_name	A descriptive name for the client instance. This name appears when the user is prompted for authorization.
token_endpoint_auth_method	The client authentication method. PingFederate accepts the following values: • `none` • `client_secret_basic` • `client_secret_post` • `tls_client_auth` For more information, see Mutual TLS Profiles for OAuth clients. • `private_key_jwt` For more information, see Client Authentication. • `client_secret_jwt` For more information, see Client Authentication.
tls_client_auth_subject_dn	The subject DN of the client certificate. This field is required if `tls_client_auth` is the value of the `token_endpoint_auth_method` parameter.

Metadata field	Metadata description
token_endpoint_auth_signing_alg	The signing algorithm that the client must use to sign the JSON Web Token (JWT) for client authentication. This field applies only when the `token_endpoint_auth_method` parameter is provided with a value of `private_key_jwt` or `client_secret_jwt`. For `private_key_jwt`, PingFederate accepts the following values: • `RS256` - RSA using SHA-256 • `RS384` - RSA using SHA-384 • `RS512` - RSA using SHA-512 • `ES256` - ECDSA using P256 Curve and SHA-256 • `ES384` - ECDSA using P384 Curve and SHA-384 • `ES512` - ECDSA using P521 Curve and SHA-512 • `PS256` - RSASSA-PSS using SHA-256 • `PS384` - RSASSA-PSS using SHA-384 • `PS512` - RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment, or an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. For `client_secret_jwt`, PingFederate accepts the following values: • `HS256` - HMAC using SHA-256 • `HS384` - HMAC using SHA-384 • `HS512` - HMAC using SHA-512 If this parameter is not provided, the client can use any of the supported signing algorithms.

Metadata field	Metadata description
request_object_signing_alg	The signing algorithm that the client must use to sign its request objects for transmission of request parameters. Applicable only when the client might send its authorization requests using request objects. PingFederate accepts the following values: • `RS256` - RSA using SHA-256 • `RS384` - RSA using SHA-384 • `RS512` - RSA using SHA-512 • `HS256` - HMAC using SHA-256 • `HS384` - HMAC using SHA-384 • `HS512` - HMAC using SHA-512 • `ES256` - ECDSA using P256 Curve and SHA-256 • `ES384` - ECDSA using P384 Curve and SHA-384 • `ES512` - ECDSA using P521 Curve and SHA-512 • `PS256` - RSASSA-PSS using SHA-256 • `PS384` - RSASSA-PSS using SHA-384 • `PS512` - RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment, or an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. When this parameter is not provided, the client can use any of the supported signing algorithms. For more information about request objects, see RFC 9101: JWT Secured Authorization Request (JAR).

Metadata field	Metadata description
jwks_uri, and jwks	The URL of the JSON Web Key Set (JWKS) or the actual JWKS from the client. If the client is configured to use the [.codeph]``private_key_jwt`` or [.codeph]``client_secret_jwt`` client authentication method, to transmit request parameters in signed request objects, or to transmit CIBA request parameters in signed request objects, only one of the previous values is required for {pingfed} to verify the authenticity of the JWTs. Either value can be defined even if the client is not configured to use JWTs for authentication or transmission of request parameters. This flexibility allows the client to transmit request parameters in signed request objects for some requests and without the use of signed request objects for some other transactions. For information on runtime processing, see Authorization endpoint. If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment, or integrated with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. If the client is configured to encrypt ID tokens using an asymmetric encryption algorithm, either the JWKS URL or the actual JWKS must be provided. See the **ID Token Key Management Encryption Algorithm** setting.
redirect_uris	An array of one or more redirection URIs where the OAuth AS may redirect the resource owner's user agent after authorization is obtained. The authorization code and implicit grant types require at least one redirection URI.
logo_uri	The location of the logo used on user-facing OAuth grant authorization and revocation pages. For best results with the installed HTML templates, the recommended size is 72 x 72 pixels.
scope	A space-separated list of one or more scopes, which a client can request.

Metadata field	Metadata description
grant_types	An array of one or more grant types, which a client can request. PingFederate accepts the following values: - `authorization_code` - `implicit` - `refresh_token` - `client_credentials` - `urn:ietf:params:oauth:grant-type:device_code` - `urn:openid:params:grant-type:ciba` - `password` - `extension` (JWT Bearer Token or SAML 2.0 Bearer Assertion) For more information about each grant type, see Grant types.

<table>
<tr><th>Metadata field</th><th>Metadata description</th></tr>
<tr><td>response_types</td><td>An array of one or more response types, which a client can request.
PingFederate accepts the following values:
• `code`
• `code id_token`
• `code id_token token`
• code token``
• `id_token`
• `id_token token`
For more information about these response types, see Definitions of Multiple-Valued Response Type Combinations.
If one or more response types are specified, the resulting client is only allowed to send one of the specified response types at runtime. Requests from this client with other response types will be rejected.
Response type and grant type parameters must be provided in tandem because certain response types require one or more grant types, and vice versa. The following table provides a summary of their relationship.
<table>
<tr><th>response type</th><th>grant types</th></tr>
<tr><td>`code`</td><td>`authorization_code`</td></tr>
<tr><td>`code id_token`</td><td>`authorization_code` and `implicit`</td></tr>
<tr><td>`code id_token token`</td><td>`authorization_code` and `implicit`</td></tr>
<tr><td>`code token`</td><td>`authorization_code` and `implicit`</td></tr>
<tr><td>`id_token`</td><td>`implicit`</td></tr>
<tr><td>`id_token token`</td><td>`implicit`</td></tr>
<tr><td>`token`</td><td>`implicit`</td></tr>
</table>
</td></tr>
</table>

Metadata field	Metadata description
id_token_signed_response_alg	The JSON Web Signature (JWS) algorithm required for the OpenID Connect (OIDC) tokens. Allowed values: - `none` - No signing algorithm - `HS256` - HMAC using SHA-256 - `HS384` - HMAC using SHA-384 - `HS512` - HMAC using SHA-512 - `ES256` - ECDSA using P256 Curve and SHA-256 - `ES384` - ECDSA using P384 Curve and SHA-384 - `ES512` - ECDSA using P521 Curve and SHA-512 - `RS256` - RSA using SHA-256 - `RS384` - RSA using SHA-384 - `RS512` - RSA using SHA-512 - `PS256` - RSASSA-PSS using SHA-256 - `PS384` - RSASSA-PSS using SHA-384 - `PS512` - RSASSA-PSS using SHA-512 **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment, or an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. **Important** If static keys for OAuth and OpenID Connect are enabled, use either an RSA algorithm or an EC algorithm that has been configured with an active static key.

Metadata field	Metadata description
id_token_encrypted_response_alg	The algorithm used to encrypt or otherwise determine the value of the content encryption key. Allowed values: • `dir` - Direct Encryption with symmetric key • `A128KW` - AES-128 Key Wrap • `A192KW` - AES-192 Key Wrap • `A256KW` - AES-256 Key Wrap • `A128GCMKW` - AES-GCM-128 key encryption • `A192GCMKW` - AES-GCM-192 key encryption • `A256GCMKW` - AES-GCM-256 key encryption • `ECDH-ES` - ECDH-ES • `ECDH-ES+A128KW` - ECDH-ES with AES-128 Key Wrap • `ECDH-ES+A192KW` - ECDH-ES with AES-192 Key Wrap • `ECDH-ES+A256KW` - ECDH-ES with AES-256 Key Wrap • `RSA-OAEP` - RSAES-OAEP • `RSA-OAEP-256` - RSAES OAEP using SHA-256 and MGF1 with SHA-256
id_token_encrypted_response_enc	The content encryption algorithm used to perform authenticated encryption on the plain text payload of the token. Required if an algorithm is provided through the **`id_token_encrypted_response_alg`** parameter. Allowed values: • `dir` - Direct Encryption with symmetric key • `A128CBC-HS256` - Composite AES-CBC-128 HMAC-SHA-256 • `A192CBC-HS384` - Composite AES-CBC-192 HMAC-SHA-384 • `A256CBC-HS512` - Composite AES-CBC-256 HMAC-SHA-512 • `AES-GCM-128` - A128GCM • `AES-GCM-192` - A192GCM • `AES-GCM-256` - A256GCM

Metadata field	Metadata description
introspection_signing_alg_values (optional)	The JSON Web Signature (JWS) algorithm used to sign the token introspection response. Allowed values: • `HS256` HMAC using SHA-256 • `HS384` - HMAC using SHA-384 • `HS512` - HMAC using SHA-512 • `RS256` - RSA using SHA-256 • `RS384` - RSA using SHA-384 • `RS512` - RSA using SHA-512 • `ES256` - ECDSA using P256 Curve and SHA-256 • `ES384` - ECDSA using P384 Curve and SHA-384 • `ES512` - ECDSA using P521 Curve and SHA-512 • `PS256` - RSASSA-PSS using SHA-256 • `PS384` - RSASSA-PSS using SHA-384 • `PS512` - RSASSA-PSS using SHA-512 **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment, or an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. The default value is `RS256`. **None** is not an allowed value.

Metadata field	Metadata description
introspection_encryption_alg_values (optional)	The JSON Web Encryption (JWE) algorithm used to encrypt the content-encryption key of the token introspection response. Allowed Values: • `DIR` - Direct Encryption with symmetric key • `A128KW` - ES-128 Key Wrap • `A192KW` - AES-192 Key Wrap • `A256KW` - AES-256 Key Wrap • `A128GCMKW` - AES-GCM-128 key encryption • `A192GCMKW` - AES-GCM-192 key encryption • `A256GCMKW` - AES-GCM-256 key encryption • `ECDH_ES` - ECDH-ES • `ECDH_ES_A128KW` - ECDH-ES with AES-128 Key Wrap • `ECDH_ES_A192KW` - ECDH-ES with AES-192 Key Wrap • `ECDH_ES_A256KW` - ECDH-ES with AES-256 Key Wrap • `RSA_OAEP` - RSAES-OAEP • `RSA_OAEP_256` - RSAES-OAEP using SHA-256 If asymmetric type, a JWKS or JWKS URL is required and must be valid. If symmetric type, the reversible secret is required.
introspection_encryption_enc_values	The JSON Web Encryption (JWE) content-encryption algorithm for the token introspection response. Allowed values: • `AES_128_CBC_HMAC_SHA_256` - Composite AES-CBC-128 HMAC-SHA-256 • `AES_192_CBC_HMAC_SHA_384` - Composite AES-CBC-192 HMAC-SHA-384 • `AES_256_CBC_HMAC_SHA_512` - Composite AES-CBC-256 HMAC-SHA-512 • `AES_128_GCM` - Composite A128GCM • `AES_192_GCM` - Composite A192GCM • `AES_256_GCM` - Composite A256GCM **Note** This field is required if `introspection_signing_alg_values_supported` is specified. This field must be empty if `introspection_signing_alg_values_supported` is not specified.

Metadata field	Metadata description
backchannel_token_delivery_mode	The token delivery method that the client supports. PingFederate supports poll and ping. Set to `poll` if the client can check for the authorization results periodically at the token endpoint. Set to `ping` if the client prefers to wait for a ping callback message from PingFederate as a signal that the authorization result is ready for pickup. If this parameter is not provided and the CIBA grant type is enabled, the poll method is assumed.
backchannel_client_notification_endpoint	The client's notification endpoint, to which PingFederate sends its ping call back messages. Required only if `ping` is the configured token delivery method.
backchannel_authentication_request_signing_alg	The signing algorithm that the client must use to sign its request objects for transmission of request parameters. PingFederate accepts the following values: • `RS256` - RSA using SHA-256 • `RS384` - RSA using SHA-384 • `RS512` - RSA using SHA-512 • `HS256` - HMAC using SHA-256 • `HS384` - HMAC using SHA-384 • `HS512` - HMAC using SHA-512 • `ES256` - ECDSA using P256 Curve and SHA-256 • `ES384` - ECDSA using P384 Curve and SHA-384 • `ES512` - ECDSA using P521 Curve and SHA-512 • `PS256` - RSASSA-PSS using SHA-256 • `PS384` - RSASSA-PSS using SHA-384 • `PS512` - RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment, or an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. If this parameter is not provided and the CIBA grant type is enabled, the client can use any of the allowed signing algorithms.

Metadata field	Metadata description
backchannel_user_code_parameter	Indicates whether the client supports user code. The purpose of this code is to authorize the transmission of an authentication request to the user's authentication device. A valid value is either `true` or `false`. If this parameter is not provided and the CIBA grant type is enabled, user code support is not enabled. **Note** When user code support is enabled, the associated CIBA request policy must also be user code enabled.
sector_identifier_uri	A URL using the HTTPS scheme that references a JSON file containing an array of `redirect_uri` values. For more information, see .net/specs/openid-connect-registration-1_0.html//[].
subject_type	The type of subject used by the sector identifier, such as `public` or `pairwise`.

Related links

- OAuth 2.0 Dynamic Client Registration Protocol, Client Metadata (tools.ietf.org/html/rfc7591)
- OpenID Connect Dynamic Client Registration, Client Metadata (openid.net/specs/openid-connect-registration-1_0.html#ClientMetadata)
- Managing client configuration defaults

Configuring scope constraints

On the **Scope Constraints** tab, you can configure which scopes or scope groups that developers can request when registering clients using dynamic client registration.

About this task

All clients created through dynamic client registration share this configuration. If a certain client requires a different set of common scopes, exclusive scopes, or both, modify the client configuration using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created. Scopes can also be overridden by client registration policies enforced during dynamic client registration.

Steps

1. Go to **System > OAuth Settings > Client Settings** and click **Scope Constraints**.
2. To restrict clients created with the Dynamic Client Registration protocol to a subset of common scopes, select the **Restrict Common Scopes** check box and one or more applicable common scopes.

   Result:

Your selections impact the developers in several ways:

- If you do not select the **Restrict Common Scopes** check box, developers can send client registrations without including the desired scopes. If the requests are valid, the clients are configured with all the common scopes and scope groups.
- If you select the **Restrict Common Scopes** check box without selecting at least one common scope or scope group, clients resulting from valid client registrations are configured without any common scopes or scope groups.
- If you select the **Restrict Common Scopes** check box with one or more applicable common scopes or scope groups, developers must send client registrations with the desired common scopes and scope groups. Otherwise, clients resulting from otherwise valid requests are also configured without any common scopes or scope groups.

3. To allow clients created with the Dynamic Client Registration protocol to request for a subset of exclusive scopes, select one or more applicable exclusive scopes in the **Allowed Exclusive Scopes** field.

   Result:

   Your selections impact the developers in several ways:

   - If you do not select any exclusive scope, clients resulting from valid client registrations are configured without any exclusive scopes or scope groups.
   - If you select one or more applicable exclusive scopes or scope groups, developers must send client registrations with the desired exclusive scopes and scope groups. If they fail to do so, clients resulting from otherwise valid requests are also configured without any exclusive scopes or scope groups.

Result

Restricting common scopes and allowing exclusive scopes are not mutually exclusive. You can configure both options based on your use cases.

If you configure both options, developers must send client registrations with the desired common and exclusive scopes.

Depending on the configured dynamic scope patterns and whether they are defined as common or exclusive dynamic scopes, this configuration can impact the results of scope evaluation. The default scope is always available to all clients. For more information, see the Dynamic scope evaluation and per-client scope management section in Scopes and scope management.

Related links

- Managing OAuth clients
- PingFederate administrative API
- OAuth Client Management Service

Managing client configuration defaults

On the **Client Configuration Defaults** tab, specify the default settings that are proprietary to PingFederate for clients created with the OAuth 2.0 Dynamic Client Registration protocol.

About this task

Although these settings are shared among all clients created through dynamic client registration, they can be overridden by client registration policies enforced during dynamic client registration. You can also modify the client configuration using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created.

Steps

1. Go to **System > OAuth Settings > Client Settings** and click **Client Configuration Defaults**.
2. **Optional:** Modify the default values as needed.

   The following table describes each field.

Field	Description
Private Key JWT - Replay Prevention	This field determines whether PingFederate mandates [.ph]a unique signed JWT from the client for each request when the client is configured to authenticate using the private_key_jwt client authentication method, to transmit request parameters using signed request objects, or to do both. This check box is not selected by default. **Note** The underlying Assertion Replay Prevention Service is cluster-aware. For more information, see Assertion Replay Prevention Service.
Require Signed Requests	Determines whether the client must transmit request parameters in a single, self-contained parameter. The parameter name is `request`. The value of the `request` parameter is [.ph]a signed JWT whose claims represent the authorization request parameters. The OpenID Connect (OIDC) specification calls this JWT a request object. This check box is not selected by default.
Require JWT Secured Authorization Response Mode (JARM)	When selected, the client must use JARM. The client's authorization requests must include one of the following authorization response mode values: ◦ `jwt`: ■ query JWT response in the case of the authorization code grant ■ fragment JWT response in the case of the implicit grant ◦ `query.jwt`: ■ JWT response in the case of the authorization code grant ■ Failure in the case of the implicit grant unless the response is an encrypted JWT based on the client settings ◦ `fragment.jwt`: fragment JWT response ◦ `form_post.jwt`: auto form-post JWT response JARM is a mechanism to enhance the security of the standard authorization response. It adds support for signing and encryption, sender authentication, and audience restriction. It also offers protection from replay, credential leakage, and mix-up attacks. JARM can be combined with any response type. For more information, see the JARM specification. This setting works in conjunction with the other **JWT Secured Authorization Response Mode** settings on this window. This check box is cleared by default.

Field	Description
Default Access Token Manager	Determines the default Access Token Management (ATM) instance for this client.
Restrict to Default Access Token Manager	If selected, the client can use only the **Default Access Token Manager**, even if other ATMs include the client in their access control lists. This check box is not selected by default.
Persistent Grants Max Lifetime	Overrides the **Persistent Grant Max Lifetime** value set globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Grants Do Not Expire** ◦ A custom value in days, hours, or minutes. ⓘ **Note** This setting can be overridden per grant-mapping configuration through the use of an extended persistent grant attribute `PERSISTENT_GRANT_LIFETIME`. The `PERSISTENT_GRANT_LIFETIME` attribute is defined in **System > OAuth Settings > Authorization Server Settings**. When this attribute is active, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions in individual grant-mapping configurations. For grant-mapping configurations that do not require this fine-grain control, you can configure them to use the default value.
Persistent Grants Idle Timeout	Overrides the **Persistent Grant Idle Timeout** field value set globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Grants Do Not Timeout Due To Inactivity** ◦ A custom value in days, hours, or minutes If you configure an idle timeout value, the idle timeout window slides when a persistent grant updates. When you have an idle timeout value configured without a maximum lifetime, persistent grants remain valid until they expire because of inactivity or until the grant storage revokes or removes them. When you have an idle timeout value configured with a maximum lifetime, persistent grants remain valid until they expire due to inactivity or lifetime expiration or until the grant storage removes them.
Client Authentication Certificate Issuer DN	Select a trusted CA from the list. You can review CA certificates imported into PingFederate them in **Security > Certificate & Key Management > Trusted CAs**. You can select **Trust Any** to trust all the issuers found in the list. The default selection is **None (Client TLS Certificate Authentication Disabled)**, which does not allow developers to submit client registrations with a `token_endpoint_auth_method` parameter value of `tls_client_auth`.

Field	Description
Refresh Token Rolling Policy	Overrides the **Roll Refresh Token Values** setting configured globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Roll** This value does not override the **Minimum Interval to Roll Refresh Tokens** value set in the **Authorization Server Settings** window. ◦ **Don't Roll**
Refresh Token Rolling Interval	Overrides the **Minimum Interval to Roll Refresh Tokens** value set in the **Authorization Server Settings** window. Select one of the following options: ◦ **Use Global Setting** (default) ◦ A custom value in hours. The maximum is `8760` hours (365 days).
Refresh Token Rolling Grace Period (seconds)	The amount of time in seconds that a rolled refresh token is still valid in the event that the client failed to receive an updated one during a roll. Select one of the following options: ◦ **Use Global Setting** (default) ◦ A custom value in seconds. The maximum is `86400`. For security reasons, we recommend setting the custom value as low as possible. ⓘ **Note** If specified: ◦ the currently issued refresh tokens must be refreshed once and converted to the new format. ◦ this overrides the **Authorization Server Settings** default setting.

Field	Description
OpenID Connect	***ID Token Signing Algorithm*** Select the signing algorithm for the ID tokens from the list. The default algorithm is **RSA using SHA-256**. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect. ⓘ **Note** If static keys for OAuth and OpenID Connect are enabled, Elliptic-curve cryptography (EC) algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration might affect runtime transactions and require additional changes here. For more information, see Keys for OAuth and OpenID Connect. ⓘ **Note** While all settings on this window can be overridden by client registration policies enforced during the registration, **ID Token Signing Algorithm** is the only default setting that can also be overridden by including a different `id_token_signed_response_alg` client metadata value in the client registration. For a list of supported signing algorithm, see the `id_token_signing_alg_values_supported` parameter values returned by the PingFederate OpenID Provider configuration endpoint at `/.well-known/openid-configuration`. ***Policy*** Select a specific OpenID Connect policy from the list.

Field	Description
Device Authorization	Determines whether to use global device authorization grant settings defined in **System > OAuth Settings > Authorization Server Settings**. The default selection is **Use Global Settings**. You can select **Override** and configure any of the following settings. ***User Authorization URL*** This field determines whether PingFederate should use a different URL when formulating the verification URLs to be included in its device authorization responses. For more information, see Device authorization endpoint.For example, if this field is configured with a value of `https://www.example.org/welcome`, PingFederate returns https://www.example.org/welcome and https://www.example.org/welcome?user_code=<*activationcode*> as the verification URIs.After processing the device authorization response, which includes the verification URIs, the device presents one of them to the user. The user is expected to browse to the presented verification URI on a second device. **Important** The target web server must redirect the browser to PingFederate at its user authorization endpoint. For more information, see User authorization endpoint. It must also preserve the `user_code` parameter value, if provided. For example, if your PingFederate server's base URL is https://www.example.com and the **User Authorization URL** value is `https://www.example.org/welcome`, the target web server must redirect as follows: ◦ https://www.example.org/welcome to https://www.example.com/as/user_authz.oauth2 ◦ https://www.example.org/welcome?user_code=<*activationcode*> to https://www.example.com/as/user_authz.oauth2?user_code=<*activationcode*> This field has no default value. ***Pending Authorization Timeout (seconds)*** The lifetime of an activation code (the `user_code` parameter value) in seconds.This field has no default value. ***Device Polling Interval (seconds)*** The amount of time in seconds that the device waits between polling requests to the PingFederate token endpoint.This field has no default value. ***Bypass Activation Code Confirmation*** When PingFederate receives a verification request that includes an activation code (the `user_code` parameter value), it prompts the user to confirm the activation code.For PingFederate to skip this confirmation step, select this check box.This check box is not selected by default.

Field	Description
Require Proof Key for Code Exchange (PKCE)	This field is only applicable when the client is configured to support the authorization code grant type. Determines whether the client must provide certain parameters to reduce the risk of authorization code interception attack. For more information, see the Proof Key for Code Exchange (PKCE) by OAuth Public Clients specification. When enabled, this client must include a one-time string value through the use of the `code_challenge` parameter in its authorization request. For more information, see Authorization endpoint. It must also submit the corresponding code verifier through the `code_verifier` parameter in its token request when exchanging an authorization code for an access token. For more information, see OAuth grant type parameters. This check box is not selected by default.
Polling Interval (seconds)	Specifies the number of seconds that the client must wait between its attempts to check for the authorization results at the token endpoint. When PingFederate receives a token request within this time interval, it returns a `slow_down` error message to the client. A valid value ranges from `1` to `3600`. The default value is `3`.
Policy	Specifies the CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication request, such as the maximum lifetime of authentication requests, the validity of unsigned login hint tokens, and the mapping configuration of identity hints. Select an existing CIBA policy. You may also leave the selection of **Default** to indicate that PingFederate should use the CIBA request policy that has been designated as the default CIBA request policy in **Applications > OAuth > CIBA Request Policies**.
Require CIBA Signed Requests	Determines whether the client must transmit request parameters in a single, self-contained parameter. The parameter name is `request`. The value of the `request` parameter is [.ph]a signed JWT whose claims represent the authorization request parameters. The OIDC specification calls this JWT a request object. This check box is not selected by default. Note that if client-initiated back channel authentication (CIBA) signed requests are required, the dynamic client registration must include either the JWKS URL or the actual JWKS.
Token Exchange	This field determines which token exchange processor policy PingFederate uses when the OAuth server receives an OAuth token exchange request from the client. If you select **Default**, PingFederate uses the token exchange processor policy that was set as the default on the **Token Exchange Processor Policy Management** tab, under **Applications > Token Exchange > Processor Policies**. For more information, see OAuth token exchange.

Field	Description
Client Secret Retention Period	Set the number of minutes that PingFederate retains a client secret after it is replaced. The retention period lets application teams update the client secret in their apps without disruption. If the value is `0`, PingFederate won't retain replaced secrets. The maximum value is `43800`, which is one month.Use one of these options: ◦ Select **Use Global Setting**, which is configured on the **Authorization Server Settings** window ◦ Override the global setting by entering a number in the field The default is to use the global setting. For information about configuring the global setting, see "Client Secret Retention Period" in Configuring authorization server settings. ⓘ **Note** You must select the **Retain Client Secret** check box on the **Dynamic Client Registration** page for this option to appear.

Related links

- Supported client metadata
- OpenID Connect specification, Client Authentication
- OpenID Connect specification, Passing a Request Object by Value
- OpenID Provider configuration endpoint

Selecting client registration policies

Client registration policies can provide additional control over which registrations and configurations are accepted and stored for each client created with the OAuth 2.0 Dynamic Client Registration protocol.

About this task

If multiple policies are configured, PingFederate executes all of them based on the display order. If PingFederate completes the current policy, it moves on to the next policy. Otherwise, PingFederate returns an error message to the developers.

> ⓘ **Note**
>
> PingFederate must complete all policies successfully before a client can be created with the OAuth 2.0 Dynamic Client Registration protocol.

Steps

1. Go to **System > OAuth Settings > Client Registration Policies**.
2. **Optional:** Select a Client Registration Policy instance from the **Available Policies** list and click **Add**.

> **Important**
>
> Select this option to add a layer of protection against unwanted client registrations.

If you have not yet defined the desired Client Registration Policy instance, click **Manage Client Registration Policies**.

3. **Optional:** Repeat the previous step to add other Client Registration Policy instances.

   Add as many Client Registration Policy instances as necessary. Click the up and down arrows to adjust the execution order. Use the **Delete** and **Undelete** buttons to remove an existing instance or cancel the removal request.

Related links

- Managing Client Registration Policy instances

Reviewing client settings

On the **Summary** tab, review your client settings.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Managing Client Registration Policy instances

The Client Registration Policy plugin allows you to write custom processing rules to provide additional control over which registrations and configurations are accepted and stored for each client created with the OAuth 2.0 Dynamic Client Registration protocol.

About this task

Depending on the technical requirements of your use cases, you can create Client Registration Policy plugins using the PingFederate SDK. After deploying your plugins, you can create and configure instances of them. Configuration requirements vary based on your custom solutions. When you are ready to configure dynamic client registration, add your policies to its configuration.

Steps

1. Implement the `DynamicClientRegistrationPlugin` interface.

   For more information, refer to the Javadoc for the `DynamicClientRegistrationPlugin` interface, the `SoftwareStatementValidatorPlugin.java` file for a sample implementation, and the SDK developer's guide for build and deployment information.

   > **Tip**
   >
   > The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

2. Create, modify, or remove one or more instances.

   Choose from:

   - To configure a new instance, click **Create New Instance**.
   - To modify an existing instance, select it under **Instance Name**.
   - To remove an existing instance or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

     > **Note**
     >
     > You can remove a Client Registration Policy instance only if it is not currently in-use by dynamic client registration.

   - To save the plugin configuration, click **Save**.

Result

> **Important**
>
> A Client Registration Policy instance is not enforced, or executed as part of the dynamic client registration process, until it is selected on the **Client Registration Policies** window

Related links

- Configuring dynamic client registration settings
- Selecting client registration policies

Configuring a Client Registration Policy instance

For additional control over clients created with the OAuth 2.0 Dynamic Client Registration protocol, use the **Client Registration Policies** window to create or modify a Client Registration Policy instance.

Steps

1. Go to **System > OAuth Settings > Client Registration Policies**.

   Choose from:

   - To configure a new instance, click **Create New Instance**.
   - To modify an existing instance, select it under **Instance Name**.

2. On the **Type** tab, enter a name and an ID for a new instance, and then select a plugin from the list.

   When modifying an existing policy plugin instance, you can only change the **Instance Name** field.

   If the **Type** list does not contain the desired Client Registration Policy plugin, create one using the PingFederate SDK. For more information, refer to the Javadoc for the `DynamicClientRegistrationPlugin` interface, the `SoftwareStatementValidatorPlugin.java` file for a sample implementation, and the SDK developer's guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

3. In the **Instance Configuration** tab, follow the on-screen instructions to configure the Client Registration Policy instance.

> **Note**
>
> This window varies depending on the selected Client Registration Policy plugin.

4. On the **Summary** tab, review the plugin configuration. Click **Done**.

5. In the **Client Registration Policies** window, click **Save**.

Result

> **Important**
>
> A Client Registration Policy instance is not enforced, or executed as part of the dynamic client registration process, until it is selected on the **Client Registration Policies** window

Related links

- Configuring a Response Type Constraints instance
- Selecting client registration policies

Configuring a Response Type Constraints instance

The Response Type Constraints policy plugin allows administrators to control which flows are allowed for clients created through the OAuth 2.0 Dynamic Client Registration protocol.

About this task

Configure an instance of the Response Type Constraints policy to limit which of the following `response_types` parameter values are allowed:

- **code**
- **code id_token**
- **code id_token token**
- **code token**
- **id_token**
- **id_token token**
- **token**

For more information about flows and response types, see the OpenID Connect specification.

Steps

1. Go to **System > OAuth Settings > Client Registration Policies**.

   Choose from:

   - To configure a new instance, click **Create New Instance**.
   - To modify an existing instance, select it under **Instance Name**.

2. On the **Type** tab, enter a name and an ID for a new instance, and then select **Response Type Constraints** from the **Type** list.

   When modifying an existing policy plugin instance, you can only change the **Instance Name** field.

3. On the **Instance Configuration** tab, clear the applicable check boxes to remove the unwanted response types.

   > **Note**
   >
   > All response types are allowed by default.

4. On the **Summary** tab, review the plugin configuration. Click **Done**.
5. In the **Client Registration Policy Instances** window, click **Save**.

Result

> **Important**
>
> Like other Client Registration Policy plugins, an instance of the Response Type Constraints policy plugin is not enforced, or executed as part of the dynamic client registration process, until it is selected in **System > OAuth Settings > Client Registration Policies**
>
> 1. If it is selected in the **Client Registration Policies** window, PingFederate discards all restricted response types when processing client registrations. If no response type is allowed, PingFederate rejects the registration and returns an error message to the originator.

Related links

- Selecting client registration policies
- Supported client metadata

Managing OAuth clients

An OAuth client application interacts with an OAuth authorization server to obtain the required access tokens to call OAuth-protected services at the resource server.

About this task

The **Clients** window displays 20 clients at a time. You can sort the display order by name, ID, creation date, or last modified timestamps. You can use the pagination controls to navigate through the rest of the clients or search clients by name or ID. A client is included in the search results if its name or ID is a partial, case-insensitive match to the search term. NOTE: Due to limits on query performance, when you sort by the creation or modification date, clients that are stored in a Java Database Connectivity (JDBC) datastore and have no value defined for the sort field will not be displayed.

Steps

1. To manage OAuth clients, go to **Applications** > **OAuth** > **Clients**.

Action	Steps
To add a client	Click **Add Client** and complete the configuration in the **Client** window.
To edit a recently modified client	Select the client and update the configuration in the **Client** window.
To enable or disable one or more clients	Click their toggle switches and then click **Save**.
To remove a client or cancel the removal request	Use the **Delete** and **Undelete** buttons for the applicable client and then click **Save**.

Result

PingFederate stores client records in XML files by default. On-disk storage allows you to manage clients using the administrative console and the administrative API. Client records are part of the configuration archive.

Alternatively, you can configure PingFederate to store client records externally, which allows you to manage client records through the OAuth Client Management Service or enable dynamic client registration for your partner-developers. In this case, client records are not part of the configuration archive. Instead, PingFederate stores them on a database server, a directory server, or another storage medium through the use of the PingFederate SDK. For more information, see OAuth client datastores.

Configuring OAuth clients

Use the **Client** page to control the usage and behavior of the applications requesting access to protected resources through the PingFederate OAuth authorization server (OAuth AS).

Steps

1. Go to **Applications** > **OAuth** > **Clients**.
2. Configure a new or existing OAuth client to suit your use cases.

   The following table describes each field on the **Client** page.

Field	Description
Client ID (Required)	A unique identifier that the client provides to the resource server (RS) to identify itself. This identifier is included with every request that the client makes.

Field	Description
Name (Required)	A descriptive name for the client instance. This name appears when the user is prompted for authorization. **Tip** If you want to localize the displayed name, you can enter a unique alias here. Then, use the same alias in language resource files.
Description	A description of what the client application does. This description appears when the user is prompted for authorization. **Tip** If you want to localize the displayed description, you can enter a unique alias in this field. Then, use the same alias in language resource files.

Client Authentication

The authentication method that the client uses.

None

Select this option if your use case doesn't require client authentication. This is the default selection.

> **Note**
>
> A selection other than **None** is required for any of the following use cases:
>
> - This client uses the **Client Credentials** grant type. Refer to the **Allowed Grant Types** checkboxes.
> - This client signs its ID tokens using an HMAC signing algorithm. Refer to the **ID Token Signing Algorithm** field.
> - This client can access the Session Revocation API. Refer to the **Allow Access to Session Revocation API** checkbox.
> - This client can access the Session Management API. Refer to the **Allow Access to Session Management API** checkbox.

Client Secret

Select this option for HTTP Basic authentication.

- To create a strong, random alphanumeric string or to manually enter a secret, click **Generate Secret**.
- To modify an existing secret, select the **Change Secret** checkbox. Then, click **Generate Secret** to create a strong random alphanumeric string or manually enter a secret.

Client Secret Retention Period

Select this option to set the number of minutes for which PingFederate retains a client secret after it is replaced. The retention period enables application teams to update the client secret in their app without disruption.

If the value is `0`, PingFederate won't retain replaced secrets. The maximum value is `43800`, which equates to a 1-month retention period.

Use one of the following options:

- Select **Use Global Setting**, which is configured on the **Authorization Server Settings** page.
- Override the global setting by entering a number in the field.

  The default is to use the global setting. Learn more about configuring the global setting in the "Client Secret Retention Period" section of Configuring authorization server settings.

Client TLS Certificate

- Select this option for mutual TLS certificate-based authentication. Recommended for client applications where security policies prohibit storing passwords.
  - Select a trusted CA in the **Issuer** list.

    These are CA certificates imported into PingFederate. You can review them on the **Security** > **Certificate & Key Management** > **Trusted CAs** page. Alternatively, you can select **Trust Any** to trust all the issuers found in the list.
  - Enter the client-certificate subject DN in the **Subject DN** or extract the subject DN from the certificate if the certificate is stored on an accessible file system.

> **Important**
> If choosing this option, you must configure a secondary PingFederate HTTPS port. Refer to the property `pf.secondary.https.port` in the table under Configuring PingFederate properties.

Private Key JWT

Select this option for the private_key_jwt client authentication method as defined in Client Authentication in the OpenID Connect (OIDC) (OIDC) specification.

Client Secret JWT

Select this option for the client_secret_jwt client authentication method, as defined in Client Authentication in the OpenID Connect (OIDC) specification.

> **Note**
> When authenticating an OAuth client that uses the private key JWT authentication scheme, PingFederate validates that the issuer and subject claims in the JWT have the same value.
> The following administrative API endpoint exposes the validation on/off switch:
>
> ```
> https://{{pf_base_host_port}}/pf-admin-api/v1/configStore/
> oauth-credentials-validator/issuerMustBeEqualToClientId
> ```
>
> To disable validation, send an HTTP POST request with the following body to the endpoint:
>
> ```
> {
>   "id": "issuerMustBeEqualToClientId",
>   "stringValue": "false",
>   "type":"STRING"
> }
> ```

Replay Prevention

Select this checkbox if PingFederate should mandate a unique signed JSON Web Token (JWT) from the client for each request when the client is configured to authenticate with the `private_key_jwt` `client_secret_jwt` client authentication method, transmit request parameters using in signed request objects, or do both.

This checkbox is cleared by default.

> **Note**
> The underlying Assertion Replay Prevention Service is cluster-aware. Learn more in Assertion Replay Prevention Service.

Signing Algorithm

In the **Signing Algorithm** list, select the algorithm that the client must use to sign the JWTs for client authentication.

The following only applies if your client authentication method is `private_key_jwt`.

Field	Description
	If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment or is integrated with an hardware Security Module (HSM) and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms are available for selection. Learn more about HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. PingFederate also optionally allows defining symmetric key algorithms. The default selection is **Allow Any**, which allows the client to use any of the signing algorithms from the list.
Require Pushed Authorization Requests	When selected, the client must use the Pushed Authorization Request (PAR) endpoint `/as/par.oauth2` on the AS to initiate authorization flows. When not selected, the client can use the PAR endpoint. This checkbox is cleared by default. This setting works in conjunction with the **PAR Status** setting on the AS. For example: ◦ If PAR is **Enabled** on the AS and required on the client, the client must use PAR. ◦ If PAR is **Enabled** on the AS but not required on the client, the client can use PAR. ◦ If PAR is **Required** on the AS but not required on the client, the client must use PAR. ◦ If PAR is **Disabled** on the AS and required on the client, the client can't access the AS. Do select this checkbox if PAR is disabled on the AS. Learn in Pushed authorization requests endpoint and Configuring authorization server settings.
Require JWT Secured Authorization Response Mode	When selected, the client must use JWT secured authorization response mode (JARM). The client's authorization requests must include one of the following authorization response mode values: ◦ `jwt`: ■ Query JWT response in the case of the authorization code grant ■ Fragment JWT response in the case of the implicit grant ◦ `query.jwt`: ■ JWT response in the case of the authorization code grant ■ Failure in the case of the implicit grant unless the response is an encrypted JWT based on the client settings ◦ `fragment.jwt`: Fragment JWT response ◦ `form_post.jwt`: Auto form-post JWT response JARM is a mechanism to enhance the security of the standard authorization response. It adds support for signing and encryption, sender authentication, and audience restriction. It also offers protection from replay, credential leakage, and mix-up attacks. JARM can be combined with any response type. Learn more in the JARM specification. This setting works in conjunction with the other **JWT Secured Authorization Response Mode** settings on this page. This checkbox is cleared by default.

Field	Description
Require Signed Requests	Determines whether the client must transmit request parameters in a single, self-contained parameter. The parameter name is `request`. The value of the `request` parameter is a signed JWT whose claims represent the authorization request parameters. The OIDC specification calls this JWT a request object. **Note** If a client includes in an authorization request a request parameter other than `client_id`, as a parameter outside of the signed request object and a claim inside of the signed request object, PingFederate always uses the claim value found inside the signed request object to process the request further. For the `client_id` request parameter, the values outside of the signed request object must match the claim values inside of the signed request object. If the values do not match, PingFederate returns an error message to the client. If a request parameter is found only outside of the signed request object, PingFederate ignores the request parameter and returns no error message. **Tip** Per OAuth and OpenID Connect specifications, a client must always include in an authorization request the `client_id` parameter outside of the signed request object. Learn more about request objects in RFC 9101: JWT Secured Authorization Request (JAR).
Request Object Signing Algorithm	The signing algorithm that the client must use to sign its request objects for transmission of request parameters. Applicable only when the client might send its authorization requests using request objects. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment or is integrated with an HSM and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms are available for selection. Learn more about HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. The default selection is **Allow Any**, which allows the client to use any of the signing algorithms from the list.

Field	Description
JWKS URL and **JWKS**	The URL of the JSON Web Key Set (JWKS) or the actual JWKS from the client. ◦ The JWKS is a JSON object containing a `keys` array. This array holds one or more JSON Web Keys (JWKs). For a single key, the JSON structure still includes the `keys` array. ◦ If the client is configured to use the `private_key_jwt` or `client_secret_jwt` client authentication method to transmit request parameters in signed request objects or transmit CIBA request parameters in signed request objects, only one of the previous values is required for PingFederate to verify the authenticity of the JWTs. Either value can be defined even if the client is not configured to use JWTs for authentication or transmission of request parameters. This flexibility allows the client to transmit request parameters in signed request objects for some requests and without the use of signed request objects for some other transactions. ◦ If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment, or integrated with a hardware security module (HSM) and a static-key configuration for OAuth and OIDC. ◦ If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment or integrated with an HSM and a static-key configuration for OAuth and OIDC. You can find information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. ◦ If the client is configured to encrypt ID tokens using an asymmetric encryption algorithm, either the JWKS URL or the actual JWKS must be provided. Refer to the **ID Token Key Management Encryption Algorithm** setting.
Redirection URIs	A descriptive name for the client instance. This name appears when the user is prompted for authorization. The authorization code and implicit grant types require at least one redirection URI Enter a fully qualified URL and click **Add** for each entry required. Wildcards are allowed. However, for security reasons, make the URL as restrictive as possible. For example, https://www.example.com/OAuthClientApp/callback.jsp. **Important** If more than one URI is added or a single URI uses wildcards, the authorization code grant and the token requests must contain a specific matching `redirect_uri` parameter when contacting the authorization endpoint (`/as/authorization.oauth2`) and token endpoint (`/as/token.oauth2`).
Logo URL	The location of the logo used on user-facing OAuth grant authorization and revocation pages. For best results with the installed HTML templates, the recommended size is 72 x 72 pixels.

Field	Description
Allow Authentication API Redirectless Mode	When selected, the client can initiate an authentication API OAuth flow through the authorization endpoint without needing to handle HTTP redirections. When enabling this feature, consider the following: ◦ Redirection URLs are optional but without a redirection URL, browser-based OAuth flows will not work. ◦ This flow doesn't support getting scope consent through the user-facing **Request for Approval** page. Enabling this feature automatically enables the **Bypass Authorization Approval** and **Restrict Common Scopes** feature. ◦ The client must manage the PF cookie and, if persistent authentication sessions are configured, the PF.PERSISTENT cookie. Learn more in Mobile application authentication through REST APIs.
Bypass Authorization Approval	When selected, resource-owner approval for client access is assumed, and PingFederate no longer presents to the user an authorization consent page or redirects to a trusted web application that is responsible to prompt the user for authorization for this client. For example, use this setting when you want to deploy a trusted application and authenticate end users with an identity provider (IdP) adapter or IdP connection.
Restrict Common Scopes	Controls whether all existing and future common scopes and scope groups should be made available to the client or only the select few. When selected, the administrative console displays a list of existing common scopes and scope groups. Choose the common scope and scope groups that are intended for the client. The rest and any common scopes and scope groups created in the future become invalid for the client. If the client tries to use such scopes or scope groups, it receives an `invalid_scope` error message from PingFederate. When cleared, all existing common scopes and scope groups and those created in the future are available to the client. This is the default behavior. **Note** Depending on the configured dynamic scope patterns and whether they are defined as common or exclusive dynamic scopes, this setting can impact the results of scope evaluation. The default scope, however, is always allowed for and available to all clients. You can find detailed information in the "Dynamic scope evaluation and per-client scope management" section of Scopes and scope management.

Field	Description
Exclusive Scopes	Controls whether any exclusive scopes and scope groups should be made available to the client. When selected, the administrative console displays a list of existing exclusive scopes and scope groups. Choose the exclusive scopes and scope groups that are intended for the client. The rest and any exclusive scopes and scope groups created in the future become invalid for the client. If the client tries to use such scopes or scope groups, it receives an `invalid_scope` error message from PingFederate. When cleared, no exclusive scopes and scope groups are available to the client. This is the default behavior. ⓘ **Note** Depending on the configured dynamic scope patterns and whether they are defined as common or exclusive dynamic scopes, this setting can impact the results of scope evaluation. The default scope, however, is always allowed for and available to all clients. You can find detailed information in the "Dynamic scope evaluation and per-client scope management" section of Scopes and scope management
Authorization Detail Types	If you configured PingFederate to support rich authorization requests from OAuth clients, and you want it to process authorization details from this client, select the **Allow Authorization Details** checkbox. Then, select the checkbox beside one or more of the supported authorization detail types that you want to accept from the client. By default, the **Allow Authorization Details** checkbox is cleared and the page hides the supported types. Learn more in OAuth rich authorization requests.
Allowed Grant Types	The grant types that this client can use. Select at least one of the following: ◦ **Authorization Code** ◦ **Implicit** ◦ **Refresh Token** ◦ **Client Credentials** ◦ **Device Authorization Grant** ◦ **CIBA** ◦ **Token Exchange** ◦ **Resource Owner Password Credentials** ◦ **Assertion Grants** ◦ **Access Token Validation (Client is a Resource Server)** There is no default selection. Learn more about each grant type in Grant types.

Field	Description
Restrict Response Types	Select this checkbox to limit the `response_type` parameter values that this client can use. Available response types are: ◦ **code** ◦ **code id_token** ◦ **code id_token token** ◦ **code token** ◦ **id_token** ◦ **id_token token** ◦ **token** Learn more about these response types in Definitions of Multiple-Valued Response Type Combinations. The **Restrict Response Type** checkbox is cleared by default. If selected, you must select at least one allowable `response_type` parameter value. Additionally, the **Restricted Response Types** and **Allowed Grant Types** settings must be configured in tandem because certain response types require one or more grant types and vice versa. Authorization code grant types are compatible with the following response types: ◦ **code** ◦ **code id_token** ◦ **code id_token token** ◦ **code token** Implicit grant types are compatible with the following response types: ◦ **code id_token** ◦ **code id_token token** ◦ **code token** ◦ **id_token** ◦ **id_token token** ◦ **token**
Default Access Token Manager	Determines the default Access Token Management (ATM) instance for this client.
Restrict to Default Access Token Manager	If selected, this client can use only the **Default Access Token Manager** even if other ATMs include this client in their access control lists. When this checkbox is selected, the **Validate Against All Eligible Access Token Managers** checkbox is hidden. This checkbox is cleared by default.
Validate Against All Eligible Access Token Managers	Applicable only to resource server clients. If selected, this resource server client is not required to specify the additional `access_token_manager_id` or `aud` parameters to disambiguate the ATM instance in its token validation requests. When the resource server client doesn't specify the desired ATM instance, PingFederate validates the access tokens against all eligible ATM instances. This simplifies interactions with PingAccess by avoiding the need to align resource URIs between PingAccess and PingFederate.

Field	Description
Require Proof Key for Code Exchange (PKCE)	Displayed only when the client is configured to support the authorization code grant type. Determines whether the client must provide certain parameters to reduce the risk of authorization code interception attack. Learn more in the Proof Key for Code Exchange (PKCE) by OAuth Public Clients specification. When enabled, this client must include a one-time string value through the use of the `code_challenge` parameter in its authorization request. Learn more in Authorization endpoint. It must also submit the corresponding code verifier through the `code_verifier` parameter in its token request when exchanging an authorization code for an access token. Learn more in OAuth grant type parameters. This checkbox is cleared by default.
Persistent Grants Max Lifetime	Overrides the **Persistent Grant Max Lifetime** value set globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Grants Do Not Expire** ◦ A custom value in days, hours, or minutes. This setting can be overridden per grant-mapping configuration through the use of an extended persistent grant attribute `PERSISTENT_GRANT_LIFETIME`. The `PERSISTENT_GRANT_LIFETIME` attribute is defined in **System > OAuth Settings > Authorization Server Settings**. When this attribute is active, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions in individual grant-mapping configurations. For grant-mapping configurations that don't require fine-grain control, you can configure them to use the default value. ⓘ **Note** Grants expire at the max lifetime set when they are created. When a grant is reused, the max lifetime should be reused as well. New refresh token requests only update the `accessGrantUpdated` value, and won't change the grant's max lifetime.
Persistent Grants Idle Timeout	Overrides the **Persistent Grant Idle Timeout** field value set globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Grants Do not Timeout Due To Inactivity** ◦ A custom value in days, hours, or minutes If you configure an idle timeout value, the idle timeout window slides when a persistent grant updates. When you have an idle timeout value configured without a maximum lifetime, persistent grants remain valid until they expire because of inactivity or until the grant storage revokes or removes them. When you have an idle timeout value configured with a maximum lifetime, persistent grants remain valid until they expire due to inactivity or lifetime expiration or until the grant storage removes them.

Field	Description
Reuse Existing Persistent Access for Grant Types	Overrides the **Reuse Existing Persistent Access Grants for Grant Types** setting configured globally in **System > OAuth Settings > Authorization Server Settings**. Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Override Global Setting** (select or clear individual grant types as needed) ◦ **Implicit** ◦ **Authorization Code** ◦ **Resource Owner Password Credentials** If the **Bypass Authorization Approval** client setting isn't enabled (default) when the **Implicit** checkbox is selected, PingFederate requests consent from the user once. The user isn't asked for authorization on subsequent requests until the access grant is revoked. When the **Authorization Code** checkbox is selected, the same is true if the **Bypass Authorization for Previously Approved Persistent Grants** authorization setting is enabled and the **Bypass Authorization Approval** client setting is disabled. If the **Bypass Authorization Approval** client setting is enabled, resource-owner approval for client access is always assumed.
Refresh Token Rolling Policy	Overrides the **Roll Refresh Token Values** setting configured globally in **System > OAuth Settings > Authorization Server Settings** Select one of the following options: ◦ **Use Global Setting** (default) ◦ **Roll** This value doesn't override the **Minimum Interval to Roll Refresh Tokens** value set in the **Authorization Server Settings** page. ◦ **Do Not Roll**
Refresh Token Rolling Interval	Overrides the **Minimum Interval to Roll Refresh Tokens** value set in the **Authorization Server Settings** page. Select one of the following options: ◦ **Use Global Setting** (default) ◦ A custom value in hours. The maximum is `8760` hours, which equates to a 1-year interval.
Refresh Token Rolling Grace Period (seconds)	The amount of time in seconds that a rolled refresh token is still valid if the client failed to receive an updated one during a roll. Select one of the following options: ◦ **Use Global Setting** (default) ◦ A custom value in seconds. The maximum is `86400` . For security reasons, we recommend setting the custom value as low as possible. ⓘ **Note** If specified: ◦ The currently issued refresh tokens must be refreshed once and converted to the new format. ◦ This overrides the **Authorization Server Settings** default setting.

OIDC	*ID Token Signing Algorithm* Select the signing algorithm for the ID tokens in the list. The default algorithm is **RSA using SHA-256**. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment or is integrated with an HSM and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. Learn more about HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect. **Note** If static keys for OAuth and OIDC are enabled, elliptic-curve cryptography (EC) algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration could affect runtime transactions and require additional changes here. Learn more in Keys for OAuth and OpenID Connect. *ID Token Key Management Encryption Algorithm* The algorithm used to encrypt or otherwise determine the value of the content encryption key.PingFederate supports symmetric algorithms, such as **Direct Encryption with symmetric key**, **AES ... Key Wrap**, and **AES-GCM ... key encryption**, and asymmetric algorithms, such as **ECDH-ES**, **ECDH-ES ... Key Wrap**, and **RSAES OAEP**. **Tip** If you select a symmetric algorithm, you should provide a **Client Secret**. The client secret will be used as the symmetric key even if your client doesn’t use the client secret for authentication. If you select an asymmetric algorithm, you should provide a **JWKS** or **JWKS URL** so PingFederate can find the right client key. *ID Token Content Encryption Algorithm* The content encryption algorithm used to perform authenticated encryption on the plain text payload of the token.Required if an algorithm is selected from the **ID Token Key Management Encryption Algorithm** list. *Policy* Select a specific OIDC policy in the list. *Use Pairwise Identifier* When selected, the use of pairwise pseudonymous identifiers (PPIDs) is enabled for open banking. This checkbox is cleared by default. *Sector Identifier URI* Displayed only when **Use pairwise identifier** is selected. Optionally, enter one HTTPS URI.

Field	Description
	ⓘ **Note** If the **Track User Sessions for Logout** checkbox is selected in the **Authorization Server Settings** page, the **Client** page also displays the following: ◦ **Logout Mode** list ◦ **PingAccess Logout Capable** checkbox ◦ **Front-Channel Logout URIs** field ◦ **Back-Channel Logout URI** field ***Logout Mode*** Select one of the following options: ◦ **None**: When selected, PingFederate doesn’t send logout requests or tokens to the client. ◦ **Ping Front-Channel**: When selected, PingFederate sends logout requests, using the browser, to PingAccess and additional requests to other relying parties. ◦ **OIDC Back-Channel**: When selected, PingFederate sends a logout token to the client’s **Back-Channel Logout URI**. This feature conforms to the OpenID Connect Back-Channel Logout specification. ***PingAccess Logout Capable*** When selected, PingFederate sends logout requests through the browser to an OIDC endpoint in PingAccess as part of the logout process. Learn more in OpenID Connect endpoints in the PingAccess documentation. This checkbox is cleared by default. ***Front-Channel Logout URIs*** Enter additional endpoints at the relying parties as needed. When the **Logout Mode** is set to **Ping Front-Channel**, PingFederate sends requests to these URIs through the browser as part of the logout process. The relying parties must return an image in their logout responses. Otherwise, PingFederate returns an error message or redirects to the `InErrorResource` parameter value, if specified. ***Back-Channel Logout URI*** When the **Logout Mode** is set to **OIDC Back-Channel**, PingFederate sends a logout token to the client’s **Back-Channel Logout URI**.

Field	Description
Session API Endpoints	***Allow Access to Session Revocation API*** Select this checkbox to allow this client application to add sessions to or query the revocation status through the Back-Channel Session Revocation API endpoint at `/pf-ws/rest/sessionMgmt/revokedSris`. Authentication is required. This checkbox is cleared by default. ⓘ **Note** If clients are allowed to add sessions to the revocation list, you can enable the **Check session revocation status** option in the applicable Access Token Management instances for the token validation process to consider whether a session has been added to the revocation list. ***Allow Access to Session Management API*** The session management API lets client applications get information about user sessions, extend sessions, and revoke sessions. Learn more in Session Management API by session identifiers.

Field	Description
Device Authorization Grant	Determines whether to use global device authorization grant settings defined in **System > OAuth Settings > Authorization Server Settings**. Displayed only if the **Device Authorization Grant** grant type is enabled for the client. The default selection is **Use Global Settings**. You can select **Override** and configure any of the following settings. ***User Authorization URL*** This field determines whether PingFederate should use a different URL when formulating the verification URLs to be included in its device authorization responses. Learn more in Device authorization endpoint. For example, if this field is configured with a value of `https://www.example.org/welcome`, PingFederate returns https://www.example.org/welcome and https://www.example.org/welcome?user_code=<*activationcode*> as the verification URIs. After processing the device authorization response, which includes the verification URIs, the device presents one of them to the user. The user is expected to browse to the presented verification URI on a second device. **Important** The target web server must redirect the browser to PingFederate at its user authorization endpoint. Learn more in User authorization endpoint. It must also preserve the `user_code` parameter value, if provided. For example, if your PingFederate server's base URL is https://www.example.com and the **User Authorization URL** value is `https://www.example.org/welcome`, the target web server must redirect as follows: ◦ https://www.example.org/welcome to https://www.example.com/as/user_authz.oauth2 ◦ https://www.example.org/welcome?user_code=<*activationcode*> to https://www.example.com/as/user_authz.oauth2?user_code=<*activationcode*> This field has no default value. ***Pending Authorization Timeout (seconds)*** The lifetime of an activation code (the `user_code` parameter value) in seconds.This field has no default value. ***Device Polling Interval (seconds)*** The amount of time in seconds that the device waits between polling requests to the PingFederate token endpoint.This field has no default value. ***Bypass Activation Code Confirmation*** When PingFederate receives a verification request that includes an activation code (the `user_code` parameter value), it prompts the user to confirm the activation code. For PingFederate to skip this confirmation step, select this checkbox. This checkbox is cleared by default.

CIBA	Displayed only if the **CIBA** grant type is enabled for the client. ***Token Delivery Method*** The token delivery method that the client supports. PingFederate supports poll and ping. ◦ Select **Poll** if the client can check for the authorization results at the token endpoint periodically. ◦ Select **Ping** if the client prefers to wait for a ping callback message from PingFederate as a signal that the authorization result is ready for pickup. The default selection is **Poll**. ***Notification Endpoint*** The client's notification endpoint to which PingFederate sends its ping callback messages. Required and displayed only if ping is the configured token delivery method. ***Polling Interval (seconds)*** Specifies the number of seconds that the client must wait between its attempts to check for the authorization results at the token endpoint. When PingFederate receives a token request within this time interval, it returns a `slow_down` error message to the client. A valid value is an integer between `1` - `3600`. The default value is `3`. ***Policy*** Specifies the CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication request, such as the maximum lifetime of authentication requests, validity of unsigned login hint tokens, and mapping configuration of identity hints. Select an existing CIBA policy. You can also leave the selection of **Default** to indicate that PingFederate should use the CIBA request policy that has been designated as the default CIBA request policy in **Applications > OAuth > CIBA Request Policies**. ***User Code Support*** Indicates whether the client supports user code. The purpose of this code is to authorize the transmission of an authentication request to the user's authentication device. This checkbox is cleared by default. When user code support is enabled, the associated CIBA request policy must also be user code enabled. ***Require CIBA Signed Requests*** Determines whether the client must transmit request parameters in a single, self-contained parameter. The parameter name is `request`. The value of the `request` parameter is a signed JWT whose claims represent the authorization request parameters. The OIDC specification calls this JWT a request object. This checkbox cleared by default. If CIBA signed requests are required, the client must also be configured with either the JWKS URL or the actual JWKS from the client. ***CIBA Request Object Signing Algorithm*** The signing algorithm that the client must use to sign its request objects for transmission of request parameters.

Field	Description
	If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment or is integrated with an HSM and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. Learn more about HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect. The **Allow Any** default selection allows the client to use any of the signing algorithms from the list.
Token Exchange	Displayed only if the **Token Exchange** grant type is enabled. Select the token exchange processor policy that PingFederate uses when the AS receives an OAuth token exchange request from the client. If you select **Default**, PingFederate uses the default token exchange processor policy. Learn more in OAuth token exchange.
Token Introspection	The algorithms that allow for token introspection signing and encryption. Displayed only when the **Access Token Validation (Client is a Resource Server)** grant type is enabled. ***Token Introspection Signing Algorithm*** The JSON Web Signature (JWS) algorithm used to sign the token introspection response. The default value is `RS256`. ***Token Introspection Key Management Encryption Algorithm*** The optional JSON Web Encryption (JWE) encryption algorithm used to encrypt the content-encryption key of the token introspection response. ***Token Introspection Content Encryption Algorithm*** The JWE content-encryption algorithm for the token introspection response. It's displayed only if a **Token Introspection Key Management Encryption Algorithm** is selected.
JWT Secured Authorization Response Mode (JARM)	The algorithms that allow for JARM signing and encryption. Displayed only when the **Access Code** or **Implicit** grant type is enabled. ***JARM Signing Algorithm*** The JWS algorithm used to sign the authorization response. The default value is `RS256`. ***JARM Key Management Encryption Algorithm*** The optional JWE algorithm used to encrypt the content-encryption key of the authorization response. ***JARM Content Encryption Algorithm*** The JWE content-encryption algorithm for the authorization response. It's displayed only if a **JARM Key Management Encryption Algorithm** is selected.
Demonstrating Proof of Possession (DPoP)	Select the **Require DPop** checkbox if the client must use the DPoP protocol for authentication. The protocol is specified in OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer (DPoP). DPoP is not required by default. The **Authorization Server Settings** page provides **Demonstrating Proof-of-Possession (DPoP)** settings for configuring DPoP behavior.

3. To enable or disable the client, click the toggle.
4. On the **System** tab, add, remove, or update one or more values for any extended properties defined in **System > Server > Extended Properties**.

   > **Note**
   >
   > Any values defined for these extended properties will be passed to all applicable velocity templates and as a request context parameter in the authentication API.

   Extended property values can serve as metadata. They can also help drive authentication requirements. Learn more in Extended properties.
5. Click **Save**.

Grant contract mapping

In the first stage of the OAuth attribute mapping process, PingFederate maps attributes to persistent grant contracts.

You configure PingFederate to use attributes from the following sources:

- authentication policy contracts
- authentication sources, such as IdP adapter instances and IdP connections
- password credential validator instances for resource owner credentials

Depending on the attribute source, use one of the following PingFederate windows to configure the grant contract mapping:

- **Authentication** > **OAuth** > **Policy Contract Grant Mapping**
- **Authentication** > **OAuth** > **IdP Adapter Grant Mapping**
- **Authentication** > **OAuth** > **Resource Owner Credentials Grant Mapping**

These windows also let you configure issuance criteria to control whether PingFederate fulfills the contract.

Persistent grants, and any associated attributes and their values, remain valid until the grants expire or until PingFederate explicitly revokes them or cleans them up.

For more information about the OAuth attribute mapping process, see Mapping OAuth attributes

Managing IdP adapter grant mapping

Use the **IdP Adapter Grant Mapping** to map authentication source values into persistent grants.

Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

You can optionally set up datastore queries to supplement values returned from the source.

This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

1. Go to **Authentication > OAuth > IdP Adapter Grant Mapping** and perform one of the following actions.

Action	Steps
Create a mapping	Select the source of the attributes from the list and click **Add Mapping**.
Modify an existing mapping	Select your mapping under **Mappings**.
Remove an existing mapping or cancel the removal request	Click **Delete** or **Undelete** under **Action**. ⓘ **Note** Before removing a mapping from your configuration, ensure that it's not used by your OAuth use cases. Any corresponding entries defined in **Applications > OAuth > Access Token Mapping** will also be removed.

Related links

- Mapping OAuth attributes

Configuring IdP adapter attribute sources and user lookup

You can optionally set up datastore queries to supplement values returned from the identity provider (IdP) adapter attribute source. After the configuration is complete, you can fulfill a contract or verify a condition in the Token Authorization framework using the results from the queries.

Steps

- To set up datastore queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** window to complete the setup. For configuration steps, see Datastore query configuration.

- To skip this option, click **Next**.

Fulfilling IdP adapter grant mapping

On the **Contract Fulfillment** tab, map authentication source values into persistent grants. Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

> **Important**
>
> The `USER_KEY` attribute values must be unique across all end users, because the `USER_KEY` attribute is the user identifier to store and to retrieve persistent grants. For example, the `sAMAccountName` attribute value of an end user in one domain might match that of another end user in another domain. In this case, you can map the `Subject DN` attribute to the `USER_KEY` attribute.

Steps

1. Go to **Authentication > OAuth > IdP Adapter Grant Mapping** and select your mapping, or click **Add Mapping**.
2. On the **Contract Fulfillment** tab, select a source from the **Source** list and then select or enter a value for each attribute in the contract.

   You can map each attribute from one of the following sources:

   - `Adapter`

     When selected, the associated **Value** drop-down list contains attributes configured in the IdP adapter instance.

   - **Context**

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

     > **Note**
     >
     > As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

> ⓘ **Note**
>
> If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
>
> - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
> - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
>   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
>   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
>   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
> - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

- **Extended Properties**

  Values are returned from the client record.

- **LDAP**, **JDBC**, or **Other**

  When selected, the **Value** list is populated with attributes selected from the datastore. Select the desired attribute from the list.

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

  Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  When selected, no value selection is necessary.

- **Text**

  When selected, the text you enter is used at runtime. You can mix text with references to any of the values from the SSO token, using the `$\{attribute}` syntax.

  When applicable, you can also enter values from your datastore using the `$\{ds.attribute}` syntax, where `attribute` is any attribute that you have selected from the datastore.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

3. Click **Next**.

Defining issuance criteria for OAuth IdP adapter mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Authentication > OAuth > IdP Adapter Grant Mapping**. Select an adapter instance, and then click **Issuance Criteria**.
2. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Adapter	Select to evaluate attributes from the IdP adapter instance.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. ⓘ **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

3. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**
   - **not equal to**
   - **not equal to (case insensitive)**
   - **not equal to DN**
   - **multi-value contains**
   - **multi-value contains (case insensitive)**
   - **multi-value contains DN**
   - **multi-value does not contain**
   - **multi-value does not contain (case insensitive)**
   - **multi-value does not contain DN**

4. In the **Value** field, enter the comparison value.

> ⓘ **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

5. Click **Add**.
6. (Optional) Repeat to add more criteria.
7. (Optional) If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   Learn more in Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Reviewing the IdP adapter mapping

On the **Summary** tab, review your identity provider (IdP) adapter mapping.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Configuring IdP connection grant mapping

Use this configuration to map values obtained from the single sign-on (SSO) tokens into the persistent grants. Persistent grants remain valid until the grant expires or is explicitly revoked.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

You can optionally set up datastore queries to supplement values returned from the source.

This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

1. Go to **Authentication > Integration > IdP Connections** and select an existing identity provider (IdP) connection or click **Create Connection**.
2. On the **Connection Type** tab, select the **Browser SSO Profiles** check box and the applicable protocol.
3. On the **Connection Options** window, select the **Browser SSO** check box and then select the **OAuth Attribute Mapping** check box.

   > **Tip**
   >
   > You can also select other options on the **Connection Type** and **Connection Options** tabs. If you do, you will be prompted to complete the required configuration. For simplicity, this topic only focuses on the **OAuth Attribute Mapping** configuration.

4. On the **General Info** tab, enter the required information.
5. On the **Browser SSO** tab, click **Configure Browser SSO** and follow the steps to complete the **User-Session Creation** tab.
6. On the **OAuth Attribute Mapping** tab, select the **Map directly into Persistent Grant** option, and then click **Configure OAuth Attribute Mapping** to continue.

   Alternatively, if you have mapped an authentication policy contract (APC) in **User-Session Creation > Target Session Mapping**, you can select the **Map to OAuth via Authentication Policy Contract** option, and then select the applicable APC from the list.

Related links

- Mapping OAuth attributes

Choosing an OAuth datastore

You can optionally set up OAuth datastore queries to supplement values returned from the source.

Steps

1. On the **Data Store** tab, perform one of the following actions.

   Choose from:

   - To set up datastore queries, select a datastore from the **Active Data Store** list and then click **Next**. For configuration steps, see Datastore query configuration.
   - To skip this optional configuration, select **No Data Store**, and then click **Next**.

Fulfilling OAuth attribute mapping

On the **Contract Fulfillment** tab, map authentication source values into persistent grants.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

> **Important**
>
> The `USER_KEY` attribute values must be unique across all end users because the `USER_KEY` attribute is the user identifier to store and to retrieve persistent grants. For example, if you are configuring an **OAuth Attribute Mapping** configuration on a SAML 2.0 IdP connection and the `SAML_SUBJECT` attribute uniquely identifies all end users, you can map the `SAML_SUBJECT` attribute to the `USER_KEY` attribute.

Steps

1. For each attribute, select a source from the list and then choose or enter a value.

   - **AccountLink**

     When selected, the **Value** list is populated with **Local User ID**. You can map **Local User ID** to an attribute that represents the user identifier, such as the `USER_KEY` attribute. This source appears only if you have elected to use account linking for a target session on the **Identity Mapping** window.

   - **Assertion** or **Provider Claims**

     When selected, the **Value** list is populated with attributes from the SSO token. Select the desired attribute from the list.

     For example, to map the value of `SAML_SUBJECT` from a SAML assertion as the value of the `USER_KEY` user identifier on the contract, select **Assertion** from the **Source** list and **SAML_SUBJECT** from the **Value** list.

   - **Context**

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

     > **Note**
     >
     > As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

> **Note**
>
> If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
>
> - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
> - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
>   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
>   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
>   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
> - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

- **Extended Properties**

  Values are returned from the client record.

- **LDAP**, **JDBC**, or **Other**

  When selected, the **Value** list is populated with attributes selected from the datastore. Select the desired attribute from the list.

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

  Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  When selected, no value selection is necessary.

- **Text**

  When selected, the text you enter is used at runtime. You can mix text with references to any of the values from the SSO token, using the `$\{attribute}` syntax.

  When applicable, you can also enter values from your datastore using the `$\{ds.attribute}` syntax, where `attribute` is any attribute that you have selected from the datastore.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Next**.

Defining issuance criteria for OAuth attribute mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Authentication > OAuth > IdP Adapter Grant Mapping** and select your mapping. Click the **Issuance Criteria** tab.

2. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes.

Source	Description
AccountLink	Select to evaluate the **Local User ID** value of the user. Displayed only if **Account Linking** is the selected identity mapping method. For more information, see Choosing an identity mapping method for SP SSO.
Assertion	Select to evaluate attributes from the IdP connection.

Source	Description
Assertion or **Provider Claims**	Select to evaluate attributes from the IdP connection.
Context	Select to evaluate properties returned from the context of the transaction at runtime. ⓘ **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

3. In the **Attribute Name** list, select the attribute to be evaluated.

4. In the **Condition** list, select the comparison method.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**
   - **not equal to**
   - **not equal to (case insensitive)**
   - **not equal to DN**
   - **multi-value contains**
   - **multi-value contains (case insensitive)**
   - **multi-value contains DN**
   - **multi-value does not contain**
   - **multi-value does not contain (case insensitive)**
   - **multi-value does not contain DN**

> ⓘ **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

5. In the **Value** field, enter the comparison value.

   > **Note**
   >
   > Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

   The value of this field is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code several ways, such as displaying an error message or e-mailing an administrator.

   To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

   If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.

7. **Optional:** Repeat to add more criteria.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.

   2. In the **Expression** field, enter the required expressions.

   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.

   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.

   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing the OAuth attribute mapping summary

On the **Summary** tab, review your configuration.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.

   - To keep your changes, click **Save**.

- To discard your changes, click **Cancel**.

Managing authentication policy contract grant mapping

Use the **Authentication Policy Contract Grant Mapping** window to map values obtained from the authentication policy contract into the persistent grants. Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

You can optionally set up datastore queries to supplement values returned from the source.

This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

1. Go to **Authentication > OAuth > Authentication Policy Contract Grant Mapping** and perform one of the following actions.

Action	Steps
Create a mapping	Select the source of the attributes from the list and click **Add Mapping**.
Modify an existing mapping	Select your mapping under **Mappings**.
Remove an existing mapping or cancel the removal request	Click **Delete** or **Undelete** under **Action**. ⓘ **Note** Before removing a mapping from your configuration, ensure that it is not used by your OAuth use cases. Any corresponding entries defined in **Applications > OAuth > Access Token Mapping** will also be removed.

Related links

- Mapping OAuth attributes

Configuring policy contract attribute sources and user lookup

You can optionally set up datastore queries to supplement values returned from the policy contract attribute source.

Steps

1. Go to **Authentication > OAuth > Authentication Policy Grant Mapping** and select your mapping, or click **Add Mapping**.
2. On the **Attribute Sources & User Lookup** tab, perform one of the following actions.

   Choose from:

   - To set up datastore queries, click **Add Attribute Source** and follow the steps to complete the setup. For configuration steps, see Datastore query configuration.
   - To skip this configuration, click **Next**.

Fulfilling policy contract grant mapping

On the **Contract Fulfillment** tab, map authentication source values into persistent grants.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

The `USER_NAME` attribute presents the name shown to the resource owner on OAuth user-facing pages.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

> **Important**
>
> The `USER_KEY` attribute values must be unique across all end users, because the `USER_KEY` attribute is the user identifier to store and to retrieve persistent grants. For example, if you are configuring an OAuth attribute mapping on a SAML 2.0 identity provider (IdP) connection and the `SAML_SUBJECT` attribute uniquely identifies all end users, you can map the `SAML_SUBJECT` attribute to the `USER_KEY` attribute.

Steps

1. On the **Contract Fulfillment** tab, select a source from the **Source** list, and then select or enter a value for each attribute in the contract.

   Map each attribute from one of the following sources:

   - `Authentication Policy Contract`

     Populates the associated **Value** list with attributes associated with the APC.
   - `Context`

     Values are returned from the context of the transaction at runtime.

> **Note**
>
> If `PERSISTENT_GRANT_LIFETIME` is an extended attribute in **System > OAuth Settings > Authorization Server Settings**, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions, or the per-client **Persistent Grants Max Lifetime** setting.
>
> - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
> - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
>   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
>   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
>   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
> - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.
>
> As the **HTTP Request** context value is retrieved as a Java object rather than text, OGNL expressions are ideal to evaluate and return values.

- `Extended Properties`

  Values are returned from the client record.

- `LDAP` / `JDBC` /Other (when a datastore is used)

  Values are returned from your datastore. When you make this selection, the **Value** list populates with attributes from the datastore.

- `Expression` (when enabled)

  Provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

- `No Mapping`

  Ignores the **Value** field.

- `Text`

You can enter a text value only, or you can mix text with references to the unique user ID returned from the credentials validator, using the `${attribute}` syntax. You can also enter values from your datastore, when applicable, using the `$\{ds.attribute}` syntax, where `attribute` is any of the datastore attributes you have selected.

2. Click **Next**.

Defining issuance criteria for policy contract mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria that must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or doesn't match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values matches or doesn't match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the Source list, select the attribute's source.
2. Depending on the selection, the **Attribute Name** list populates with associated attributes. Refer to the following table for more information.

Source	Description
Authentication Policy Contract	Select to evaluate attributes from the authentication policy contract.
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

3. In the **Attribute Name** list, select the attribute to be evaluated.

4. In the **Condition list**, select the comparison method.

   Available methods:

   - **equal to**
   - **equal to (case insensitive)**
   - **equal to DN**
   - **not equal to**
   - **not equal to (case insensitive)**
   - **not equal to DN**
   - **multi-value contains**
   - **multi-value contains (case insensitive)**
   - **multi-value contains DN**
   - **multi-value does not contain**
   - **multi-value does not contain (case insensitive)**
   - **multi-value does not contain DN**

   > **Note**
   >
   > The first six conditions are intended for single-value attributes. Use one of the **multi-value …** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

5. In the **Value** field, enter the comparison value.

   > **Note**
   >
   > Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. Learn more in Attribute mapping expressions.

6. In the **Error Result** field, enter a custom error message.

   The value of this field is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

   To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

   If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

7. Click **Add**.

8. (Optional) Repeat to add more criteria.

9. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   Learn more in Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. (Optional) In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. (Optional) Click **Test**, enter values in the applicable fields, and verify the results.
   6. (Optional) Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing authentication policy contract mapping

On the **Summary** tab, review your configuration.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Managing resource owner credentials grant mapping

Use the**Resource Owner Credentials Grant Mapping** to map values obtained from the password credential validator instance into the persistent grants. Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

You can optionally set up datastore queries to supplement values returned from the source. This mapping is intended for the Resource Owner Password Credential grant type.

Steps

1. Go to **Authentication > OAuth > Resource Owner Credentials Grant Mapping** and perform one of the following actions.

Action	Steps
Create a mapping	Select the source of the attributes from the list and click **Add Mapping**.
Modify an existing mapping	Select your mapping under **Mappings**.
Remove an existing mapping or cancel the removal request	Click **Delete** or **Undelete** under **Action**. **Note** Before removing a mapping from your configuration, ensure that it is not used by your OAuth use cases. Any corresponding entries defined in **Applications > OAuth > Access Token Mapping** will also be removed.

Related links

- Mapping OAuth attributes

Configuring resource owner attribute sources and user lookup

You can optionally set up datastore queries to supplement values returned from the resource owner attribute source.

Steps

1. Go to **Authentication > OAuth > Resource Owner Credentials Grant Mapping** and select your mapping, or click **Add Mapping**.

   Choose from:

   - To set up datastore queries, click **Add Attribute Source** on the **Attribute Sources and User Lookup** tab. For configuration steps, see Datastore query configuration.
   - To skip this optional configuration, click **Next**.

Fulfilling resource owner credentials grant mapping

On the **Contract Fulfillment** tab, map authentication source values into persistent grants.

About this task

The `USER_KEY` attribute is the identifier of the persistent grants.

If extended attributes are defined in **System > OAuth Settings > Authorization Server Settings**, configure a mapping for each attribute.

> **Important**
>
> The `USER_KEY` attribute values must be unique across all end users, because the `USER_KEY` attribute is the user identifier to store and to retrieve persistent grants. For example, the `sAMAccountName` attribute value of an end user in one domain might match that of another end user in another domain. In this case, you can map the `Subject DN` attribute to the `USER_KEY` attribute.

Steps

1. On the **Contract Fulfillment** tab, select a source from the **Source** list, and then select or enter a value for each attribute in the contract.

   Map each attribute from one of the following sources:

   - `Password Credential Validator`

     When selected, the associated **Value** list populates with attributes associated with the credential-validation instance.

   - `Context`

     Values are returned from the context of the transaction at runtime.

     > **Note**
     >
     > If `PERSISTENT_GRANT_LIFETIME` is an extended attribute in **System** > **OAuth Settings** > **Authorization Server Settings**, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions, or the per-client **Persistent Grants Max Lifetime** setting.
     >
     > - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
     > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
     >   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
     >   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
     >   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
     > - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field.
     >   This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.
     >
     > As the **HTTP Request** context value is retrieved as a Java object rather than text, OGNL expressions are ideal to evaluate and return values.

   - `Extended Properties`

     Values are returned from the client record.

   - `LDAP` / `JDBC` /Other (when a datastore is used)

     Values are returned from your datastore. When selected, the **Value** list is populates with attributes from the datastore.

   - `Expression` (when enabled)

Provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

- `No Mapping`

  Ignores the **Value** field.

- **Text**

You can enter a text value only, or you can mix text with references to the unique user ID returned from the credentials validator, using the `${attribute}` syntax. You can also enter values from your datastore, when applicable, using the `$\{ds.attribute}` syntax, where `attribute` is any of the datastore attributes you have selected.

2. Click **Next**.

Defining issuance criteria for resource-owner credentials mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Authentication** > **OAuth** > **Resource Owner Credentials Grant Mapping** and select your mapping, or click **Add Mapping**.
2. On the **Issuance Criteria** tab, select the attribute's source from the **Source** list.
3. In the **Source** list, select the attribute's source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Password Credential Validator	Select to evaluate attributes from the Password Credential Validator instance.

1. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.
2. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing the resource owner credentials mapping

On the **Summary** tab, review your configuration.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Token mapping

You can configure how PingFederate maps attributes to OAuth access tokens and OpenID Connect ID tokens.

To configure how PingFederate maps attributes to OAuth access tokens, go to **Applications > OAuth > Access Token Mappings**. You can map attributes to access tokens from various context sources, depending on how users or clients authenticate. This is the second stage of mapping OAuth attributes to access tokens. The first stage is mapping OAuth attributes to persistent grant contracts. For more information, see Mapping OAuth attributes and Managing access token mappings.

OpenID Connect policies include a map of attributes to ID tokens. To configure an OpenID Connect policy, go to **Applications > OAuth > OpenID Connect Policy Management**. For more information, see Configuring OpenID Connect policies.

Access token management

PingFederate supports multiple access token management (ATM) instances. You can configure different access token policies and attribute contracts for different OAuth clients. You can also control validation of access tokens to one or more resource servers.

When defining an ATM instance, you can customize various settings, including token format, lifetime, session validation settings, and attribute contract for this instance. You can also limit the ATM instance to a list of resource URIs, a set of clients in an access control list (ACL), or both.

For example, you can use the ACL to limit which clients can obtain access tokens from a particular ATM instance. You can also add a resource server client to the ACL of multiple ATMs instances, so that only the resource server client can submit token validation requests for access tokens issued by those ATM instances.

When there are multiple ATM instances, OAuth clients can specify the desired ATM instance by providing the `access_token_manager_id` ATM ID or an `aud` resource URI in their requests to the PingFederate OAuth authorization server at the /as/authorization.oauth2 authorization endpoint, the `/as/token.oauth2` token endpoint, and the `/as/introspect.oauth2` introspection endpoint. For resource server clients, you can configure on a per-client basis whether a resource server client must specify the desired ATM instance in its token validation requests at runtime. For more information, see Configuring OAuth clients.

At runtime, the PingFederate OAuth authorization server uses the following rules to determine which ATM instances to use:

1. PingFederate limits the eligible ATM instances to those that are available in the context of the request. For most requests, these are instances that have an attribute mapping defined in the **Access Token Mapping** window. For OAuth Assertion Grant requests, it is the set of instances for which a mapping is defined in the IdP connection. If configured, the ACL can also limit which ATM instances are eligible.

2. If the request comes with an `access_token_manager_id` or `aud` parameter, PingFederate uses the information to determine the applicable ATM instance.

3. If the request does not come with either parameter, for OAuth clients supporting the OpenID Connect protocol by including the `openid` scope value, PingFederate uses the ATM instance specified by the OpenID Connect policy associated with the client. For resource server clients, you can optionally configure PingFederate to use any eligible ATM instances for the purpose of token validation.

4. If the request comes with neither of the two parameters nor the `openid` scope, PingFederate uses the default ATM instance of the client if configured, or the default ATM instance defined for the installation if eligible. For token validation requests, if resource server clients do not provide either the `access_token_manager_id` or `aud` parameter in their requests and the resource server clients have not been configured to validate against any eligible ATM instances, the same logic applies.

If no match can be found in the eligible list of ATMs, PingFederate aborts the request.

Managing access token management instances

Use the **Access Token Management** window to specify how the PingFederate OAuth AS manages access tokens.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Management**.
2. In the **Access Token Management** window, choose from the following options.

Option	Description
Configure a new instance	Click **Create New Instance**
Modify an existing instance	Click the name of instance in the **Instance Name** column
View the usage of an existing instance	Click **Check Usage** in the **Action** column on the instance's row
Remove an existing instance	Click **Delete** in the **Action** column on the instance's row

Defining an access token management instance

Define your access token management instance in the **Type** tab. This capability allows you to configure different access token policies and attribute contracts for different OAuth clients. It also provides a means to control validation of access tokens to one or more resource servers.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Management** and click **Create New Instance**.
2. On the **Type** tab, enter a name in the **Instance Name** field and an ID in the **Instance ID** field.
3. From the **Type** list, select the plugin type of the access token management instance.

   The **Type** list varies depending on the plugins deployed on your server. For information about adding a customized plugin, contact the Ping Identity Support Center.
4. **Optional:** Select a **Parent Instance** from the list.Use this option when creating an instance that is similar to an existing one. The child instance inherits the configuration of its parent. You can also override one or more settings during the setup. Select the **Override ...** check box and make the adjustments as needed in one or more subsequent windows.

Configuring an access token management instance

The configuration varies depending on the type of the access token manager.

The reference token data model

Access tokens that use the reference token data model provide a reference to a set of attributes. The resource server must de-reference the access tokens for the corresponding identity and security information at the OAuth authorization server that issued them.

The reference token data model supports both adaptive clustering and directed clustering. For adaptive clustering, PingFederate, as the authorization server, shares token information across a replica set. If region identifiers are defined, PingFederate shares token information across replica sets in multiple regions. You can optionally override this default behavior in the configuration file for adaptive clustering. For directed clustering, PingFederate shares token information among all engine nodes, despite any state server or subcluster setup.

The JSON web token data model

JSON Web Token (JWT) bearer access tokens are secure and self-contained tokens. This allows the target resource server to validate the access tokens locally or to send the access tokens to PingFederate for validation.

This configuration uses either symmetric keys or asymmetric signing-certificate keys for token security. To facilitate rollover of keys when they expire, multiple entries are allowed for either signing mechanism. The JWT token data model is suitable for both standalone and clustered environments.

Reference-token management

Configuring reference token management

Steps

1. Go to **Applications > OAuth > Access Token Management** and click **Create New Instance**.
2. In the **Instance Configuration** window, modify the default values as needed.

   The following table describes each field.

Field	Description
Token Length (Required)	The number of characters that PingFederate uses to define the token reference. Increasing the length enhances token security. The default value is `28` characters. The minimum and maximum values are 22 and 256, respectively.
Token Lifetime (Required)	The amount of time in minutes that an access token is considered valid. The default value is `120` minutes.
Lifetime Extension Policy	Indicates whether PingFederate should reset the lifetime of an access token each time the token is validated, subject to the values defined in the **Maximum Token Lifetime** and **Lifetime Extension Threshold Percentage** fields. The options are: ◦ **No Extension** ◦ **Tokens Not Backed by Persistent Access Grants (Transient Grants)** ◦ **All Tokens** The default selection is **No Extension**.
Maximum Token Lifetime	Defines an absolute maximum token lifetime for use with the **Lifetime Extension Policy** setting, in minutes. When configured, the lifetime of access tokens can be extended but not beyond the configured value. Any value, if specified, must be greater than or equal to the value specified in the **Token Lifetime** field. This optional field has no default value.
Lifetime Extension Threshold Percentage (Required)	When PingFederate is deployed in a cluster and token-lifetime extension is enabled, there must be a cluster-group remote procedure call (RPC) to extend the life of a token. To limit RPC overhead, this setting suspends the calls until the remaining time is less than the chosen value, as a percentage of token lifetime. For example, if the token lifetime is 60 minutes and the **Lifetime Extension Threshold Percentage** value is `30` percent, the lifetime will not be extended until the remaining time is less than 18 minutes. This option can drastically reduce RPC traffic between nodes, while still supporting a lifetime extension policy. The default value is `30` percent.
Advanced Fields	

Field	Description
Mode for Synchronous RPC	Synchronous RPC calls occur when a node receives a verification request for a token it does not recognize, and for token issuance. When **Majority of Nodes** is selected, the server waits for the majority of recipients to respond. It also eliminates the need for a complete state synchronization at startup. When **All Nodes** is selected, the server waits for all recipients to respond. The default selection is **Majority of Nodes**.
RPC Timeout (Required)	The timeout value between cluster nodes during synchronous communication, in milliseconds. The recommended value ranges from 100 milliseconds to 1000, or 1 second. The default value is `500` milliseconds.
Expand Scope Groups	Determines whether to expand scope groups into their corresponding scopes in the access token contents and introspection response. This check box is not selected by default.

Related links

- Adaptive clustering
- Directed clustering

Configuring JSON token management

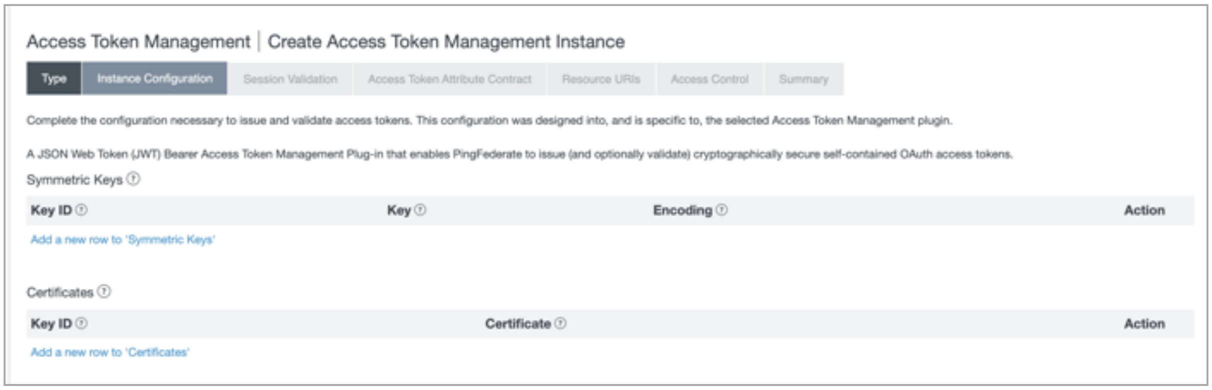
Steps

1. Go to **Applications** > **OAuth** > **Access Token Management** and click **Create New Instance**.

2. On the **Instance Configuration** tab, add one or more symmetric keys, signing certificates, or both:

   1. Click **Add a new row to...**, or click **Update** to modify an existing entry.

      > **Important**
      >
      > The **Key ID** field values must be unique across all JSON token management instances, including child instances.

      Access Token Management | Create Access Token Management Instance

      Complete the configuration necessary to issue and validate access tokens. This configuration was designed into, and is specific to, the selected Access Token Management plugin.

      A JSON Web Token (JWT) Bearer Access Token Management Plug-in that enables PingFederate to issue (and optionally validate) cryptographically secure self-contained OAuth access tokens.

      Symmetric Keys

Key ID	Key	Encoding	Action

      Add a new row to 'Symmetric Keys'

      Certificates

Key ID	Certificate	Action

      Add a new row to 'Certificates'

   2. If you have not yet created or imported your certificate into PingFederate, click **Manage Signing Certificates** to do so.

      > **Note**
      >
      > To use an RSA-based algorithm for JSON Web Signature (JWS), the key size of the signing certificate must be at least 2,048 bits. For an EC-based JWS algorithm, the key size depends on the chosen algorithm.

      Result:

      You are directed back to the **Instance Configuration** tab.

   3. Change or select the remaining field entries as needed, which the following table describes.

   For more information about JSON Web Algorithms (JWA), see the JSON Web Algorithms specification.

   +

Field	Description
Token Lifetime (Required)	The amount of time that an access token is considered valid, in minutes. The default value is `120` minutes. Additionally, you can extend the contract of the access tokens with an attribute named `exp` on the **Access Token Attribute Contract** tab. When mapping attribute values from authentication sources to the access tokens issued by this Access Token Management instance, the value you specify on the **Contract Fulfillment** tab sets the expiry as this many minutes from the current time. PingFederate internally calculates the actual epoch time for the JWT `exp` claim value.
Use Centralized Signing Key	Select this option to use a centralized key when signing JWTs using an RSA-based or EC-based algorithm. When this option is selected and static keys are not enabled, PingFederate manages and rotates a set of keys, and uses the current key corresponding to the selected signing algorithm to sign the JWT. When this option is selected and static keys are enabled, PingFederate uses the current key corresponding to the selected signing algorithm to sign the JWT. When this option is selected, an OAuth client that has been configured to use JWT access tokens through this ATM instance can retrieve the key it needs to validate the digital signature by contacting PingFederate at the `/pf/JWKS` endpoint.
JSON Web Signature (JWS) configuration	
JWS Algorithm	The hash-based message authentication code (HMAC) or the signing algorithm (EC or RSA) used to protect the integrity of the token. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or integrated with a hardware security module (HSM), additional RSASSA-PSS signing algorithms become available for selection. For more information on HSM integration, see Supported hardware security modules. Required if an asymmetric algorithm is selected in the **JWE Algorithm** list.
Active Symmetric Key ID	The ID of the symmetric key to use when producing JWTs using an HMAC-based algorithm. Required if an HMAC-based JWS algorithm is selected in the **JWS Algorithm** list.
Active Signing Certificate Key ID	The ID of the key pair and certificate to use when producing JWTs using an EC-based or RSA-based algorithm. Required if an EC-based or RSA-based JWS algorithm is selected in the **JWS Algorithm** list.
JSON Web Encryption (JWE) configuration	
JWE Algorithm	The algorithm used to encrypt or otherwise determine the value of the content encryption key. PingFederate supports symmetric algorithms, such as **Direct Encryption with symmetric key**, **AES ... Key Wrap**, and **AES-GCM ... key encryption**, and asymmetric algorithms, such as **ECDH-ES**, **ECDH-ES ... Key Wrap**, and **RSAES OAEP**.

Field	Description
JWE Content Encryption Algorithm	The content encryption algorithm used to perform authenticated encryption on the plain text payload of the token. Required if an algorithm is selected in the **JWE Algorithm** list.
Active Symmetric Encryption Key ID	The ID of the key to use when using a symmetric encryption algorithm. Required if a symmetric algorithm is selected in the **JWE Algorithm** list.
Asymmetric Encryption Key	An asymmetric encryption public key from your partner, which can be in either JSON web key (JWK) format or a certificate. Applicable only if an asymmetric algorithm is selected from the **JWE Algorithm** list. ⓘ **Note** You can only specify an asymmetric encryption key here or the partner's JSON web key set (JWKS) endpoint in the **Asymmetric Encryption JWKS URL** field.
Asymmetric Encryption JWKS URL	The HTTPS URL of a JWKS endpoint that provides a list of one or more public keys for encryption. Applicable only if an asymmetric algorithm is selected from the **JWE Algorithm** list. ⓘ **Note** You can only specify an asymmetric encryption JWK URL here or the asymmetric encryption public key from your partner in the **Asymmetric Encryption Key** field.
Enable Token Revocation	When selected, PingFederate will directly revoke a JWT access token if the client that was issued the token requests its revocation. The client can request token revocation by sending a POST request with the token to the revocation endpoint at `/as/revoke_token.oauth2`. When this check box is selected, the JWTs require a **Client ID Claim Name** and a minimum **JWT ID Claim Length** of 22 alphanumeric characters. This check box is cleared by default. ⓘ **Note** When this feature is enabled, validation of access tokens may take longer, depending on the configuration of the session revocation service.
Advanced fields	
Include Key ID Header Parameter	When selected, the key ID is used in the `kid` header parameter for the token. This check box is selected by default.
Include X.509 Thumbprint Header Parameter	When selected, the X.509 certificate thumbprint is used in the `x5t` header parameter for the token. This check box is not selected by default.

Field	Description
Default JWKS URL Cache Duration	When an asymmetric encryption JWKS URL is specified, if the remote server does not contain any cache directives in its response, PingFederate only caches the content for 720 minutes (12 hours). ⓘ **Note** When this threshold is reached or if the cache directives indicate that the content has expired at runtime, PingFederate contacts the remote server to refresh the list of encryption keys from the partner.
Include JWE Key ID header parameter	When selected, indicates whether the key ID (`kid`) header parameter will be included in the encryption header of the token, which can help identify the appropriate key during decryption. This check box is selected by default.
Include JWE X.509 Thumbprint Header Parameter	When selected, the X.509 certificate thumbprint is used as the `x5t` header parameter value in the encryption header of the token. This can help identify the appropriate key during decryption. This check box is not selected by default.
Client ID Claim Name	The name of a JWT claim used to represent the OAuth client ID. The default value is `client_id` . If the field value is empty, PingFederate will not include the client ID of the requesting client in the self-contained tokens.
Scope Claim Name	The name of a JWT claim used to represent the scope of the grant. The default value is `scope` . If the field value is empty, PingFederate will not include any scope information in the self-contained token. If clients might use the `UserInfo` endpoint to retrieve additional claims about the users, see UserInfo endpoint for more information.
Space Delimit Scope Values	When selected, indicates scope strings will be delimited by spaces rather than represented as a JSON array. This check box is not selected by default.
Issuer Claim Value	The value of the Issuer claim (`iss`) in the JWT. If left blank, this field is omitted. Additionally, you may extend the contract of the access tokens with an attribute named `iss` on the **Access Token Attribute Contract** tab. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value that you specify on the **Contract Fulfillment** tab overrides the value here.

Field	Description
Audience Claim Value	The value of the Audience claim (`aud`) in the JWT. If left blank, this field is omitted. When no value is specified, PingFederate does not validate the `aud` value, if any is included in the access token. You can also extend the contract of the access tokens with an attribute named `aud` in the **Access Token Attribute Contract** tab. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value specified in the **Contract Fulfillment** window overrides this value.
Not Before Claim Offset	To account for clock skew, you can enter a positive value that indicates the number of minutes *before* the token issuing time to which the Not Before (`nbf`) claim value will be set. For example, if the token is issued at 09:30 and the value of this offset is `10`, then the `nbf` claim value will be `09:20`. Conversely, a negative value indicates the number of minutes *after* the token issuing time to which the `nbf` claim value will be set. For example, if the token is issued at 09:30 and the value of this offset is `-10`, then the `nbf` claim value will be `09:40`. By default, the field is blank.
Include Issued At Claim	Indicates whether to include the Issued At (`iat`) claim in the JWT. By default, the check box is selected.
JWT ID Claim Length	Indicates the number of characters of the JWT ID (`jti`) claim in the JWT. The default value is `0`, meaning no claim is included.
Access Grant GUID Claim Name	The name of the JWT claim used to carry the persistent access grant GUID. If left blank, this field is omitted. If the claim is present during validation, PingFederate checks the grant database to ensure that the grant is still valid. ⓘ **Note** This use case requires that the RS must send the JWT bearer access tokens to PingFederate for validation.
JWKS Endpoint Path	The path on the PingFederate server to publish a JWKS with the keys and certificates that the partners can use for signature verification. Optional when an algorithm is selected in the **JWS Algorithm** list. If specified, the path must begin with a forward slash, such as `/oauth/jwks`. The resulting URL is https://<*pf_host*>:<*pf.https.port*>/ext/<*JWKS Endpoint Path*>. The path must be unique across all plugin instances, including any child instances.
JWKS Endpoint Cache Duration	Informs the clients of the duration that they could cache the content from the JWKS endpoint path. Applicable only if the **JWKS Endpoint Path** field is configured. The default is `720` minutes, or 12 hours.
Publish Key ID X.509 URL	Indicates whether certificates will be made accessible by the key ID at https://<*pf_host*>:<*pf.https.port*>/ext/oauth/x509/kid?v=<*id*>. This check box is not selected by default.

Field	Description
Publish Thumbprint X.509 URL	Indicates whether certificates will be made accessible by thumbprint at https://*<pf_host>*:*<pf.https.port>*/ext/oauth/x509/x5t?v=*<base64url encoded SHA-1 thumbprint>*. This check box is not selected by default.
Expand Scope Groups	Determines whether to expand scope groups into their corresponding scopes in the access token contents and introspection response. This check box is not selected by default.
Type Header Value	Indicates the value of the Type (`typ`) header in the JWT. If you do not specify a header, it is omitted.

Managing session validation settings

When an OAuth client presents an access token for validation, PingFederate acts as an OAuth authorization server and checks the expiration and the other aspects of the access token. If the validation fails, PingFederate returns an `invalid_grant` error to the client.

Before you begin

The session validation features require authentication sessions. You must enable authentication sessions for either all authentication sources or the authentication source associated with the OAuth use cases.

About this task

Edit the session validation settings on the **Session Validation** tab.

When PingFederate authentication sessions are enabled, you can optionally configure the access token validation process to evaluate the authentication sessions of the users, or resource owners, before returning the validation results to the clients. Depending on the features selected on the **Session Validation** tab, PingFederate might return an `invalid_grant` error if the associated authentication session has timed out, expired, is not found, or has been revoked.

You can also configure PingFederate to extend the authentication sessions upon successful validations.

When any session validation features are enabled, the associated session identifier (`pi.sri`) becomes available through the access tokens. For reference-style access tokens, PingFederate returns the associated session identifier in the response if the access token is valid. For JSON Web Token (JWT)-based access tokens, the session identifier is part of the access token. Through the session identifier, an OAuth client can contact the Session Management API and Session Revocation API endpoints to query the status of an authentication session, or to extend or revoke an authentication session.

The session validation features let you combine the status of access tokens and user authentication sessions. Because you can independently enable each feature per access token management (ATM) instance, you can customize unique API and web single sign-on (SSO) behaviors for your OAuth clients and users.

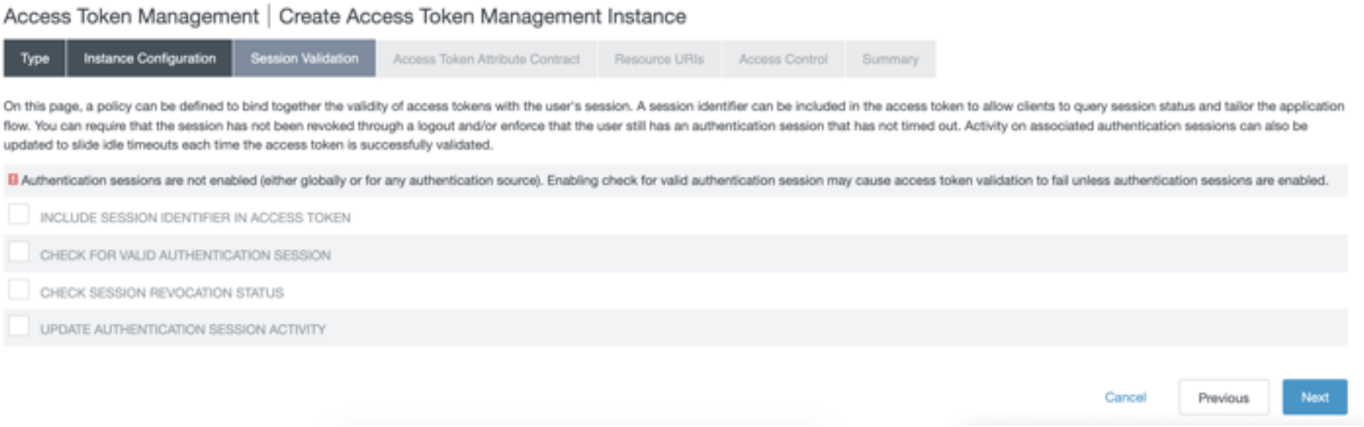
Steps

1. Go to **Applications > OAuth > Access Token Management**.
2. Select the applicable ATM instance or click **Create New Instance**.

   If you are creating a new ATM instance, complete the required fields in the **Type** and **Instance Configuration** tabs.
3. On the **Session Validation** tab, select the check box for each relevant feature.

   > **Important**
   >
   > If authentication sessions are not enabled, you can still select features on this tab, but access token validation might fail.

   If this is a child ATM instance, select the **Override Session Validation Settings** check box and edit as needed.



Access Token Management | Create Access Token Management Instance

On this page, a policy can be defined to bind together the validity of access tokens with the user's session. A session identifier can be included in the access token to allow clients to query session status and tailor the application flow. You can require that the session has not been revoked through a logout and/or enforce that the user still has an authentication session that has not timed out. Activity on associated authentication sessions can also be updated to slide idle timeouts each time the access token is successfully validated.

Authentication sessions are not enabled (either globally or for any authentication source). Enabling check for valid authentication session may cause access token validation to fail unless authentication sessions are enabled.

□ INCLUDE SESSION IDENTIFIER IN ACCESS TOKEN

□ CHECK FOR VALID AUTHENTICATION SESSION

□ CHECK SESSION REVOCATION STATUS

□ UPDATE AUTHENTICATION SESSION ACTIVITY

Each feature is independent of each other. The following table describes each feature.

Feature	Description
Include session identifier in access token	When selected, the ATM instance includes the value of the pi.sri session identifier in the access tokens it issues. When processing a refresh-grant token request through an ATM instance where this feature is enabled, PingFederate includes the same session identifier if the original access token includes one. An OAuth client that is allowed to access the session management API can get information about sessions associated with the session identifier. The client can also request PingFederate to extend or revoke the sessions. For more information, see Session Management API by session identifiers.

Feature	Description
Check for valid authentication session	When selected, an access token is considered invalid unless the user has a valid authentication session. If the user does not have a valid session, PingFederate returns an `invalid_grant` error. An authentication session is invalid when one of the following conditions applies: ◦ The authentication session has timed out based on the **Idle Timeout** field value in **Authentication > Policies > Sessions**. ◦ The authentication session has expired based on the **Max Timeout** field value in **Authentication > Policies > Sessions**. ◦ The authentication session is not found, such as if the user has logged out. You can also optimize the access token lifetime. **Note** If this ATM instance issues internally managed reference tokens, match the **Token Lifetime** value in the **Instance Configuration** tab to the **Idle Timeout** value. If you specify a **Maximum Token Lifetime** value on the **Instance Configuration** tab, ensure that the value matches that of the **Max Timeout** field. If this ATM instance issues JWT-based access tokens, match value of the **Token Lifetime** field to that of the **Max Timeout** field.
Check session revocation status	When selected, PingFederate verifies whether the session identifier has been added to the revocation list. If the session has been revoked, PingFederate returns an `invalid_grant` error. An authentication session can be revoked through the front-channel or the back-channel.
Update authentication session activity	When selected, if the access token is valid, PingFederate also extends the lifetime of the authentication session by the **Idle Timeout** field value in **Authentication > Policies > Sessions**. For externally stored authentication sessions, this operation only sends updates to the external storage when the remaining idle timeout window is less than 75%.

Related links

- Sessions
- OAuth client session management
- Introspection endpoint

Defining the access token attribute contract

On the **Access Token Attribute Contract** tab, define the attribute contract for the access tokens issued by this access token management (ATM) instance.

About this task

You must enter at least one attribute. For auditing purposes, an attribute can be chosen as the subject.

Steps

1. Go to **Applications > OAuth > Access Token Management** and select your instance, or click **Create New Instance**.
2. On the **Access Token Attribute Contract** tab use the **Extend the Contract** field and the **Add** button to add one or more attributes.
   - To always return this array in a token response, select the **Multi-Valued** check box.

     For JSON web token (JWT) bearer access tokens, you can extend the attribute contract with the following attributes.

Attribute	Description
`iss`	Adds the Issuer claim (`iss`) to the access token. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value specified on the **Access Token Attribute Contract** tab overrides any **Issuer Claim Value** defined on the **Instance Configuration** tab.
`aud`	Adds the Audience claim (`aud`) to the access token. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value you specify on the **Access Token Attribute Contract** tab overrides any **Audience Claim Value** defined on the **Instance Configuration** tab.
`exp`	Extends the value of the Expire claim (`exp`) by the specified value in minutes. ⓘ **Note** Define the Expire claim with the **Token Lifetime** setting in the **Instance Configuration** tab.
The **Client ID Claim Name** field value, the **Scope Claim Name** field value, or the **Access Grant GUID Claim Name** field value defined on the **Instance Configuration** tab of this ATM instance.	When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the values defined in the **Access Token Attribute Contract** tab override the value of the client ID, the scope, or the persistent access grant GUID.

3. Select an attribute from the **Subject Attribute Name** list.

   Result:

   When recording OAuth transactions in the audit log, PingFederate populates the subject field with values from this attribute specifically for token introspection and token validation using the `validate_bearer` grant type.

Managing resource URIs

When sending its request to the authorization endpoint on the PingFederate OAuth authorization server, an OAuth client can optionally include the requested resource in the `aud` query parameter.

About this task

If the client is sending a token exchange request, it can specify an access token manager in the `resource` query parameter.

You can specify a list of resource URIs that PingFederate OAuth authorization server can use to select this access token management instance when the `aud` or `resource` query parameter is provided.

> **Important**
>
> The resource URIs must correspond to the resource that the resource server expects.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Management** and select your instance, or click **Create New Instance**.
2. On the **Resource URI** tab, perform one of the following actions.

Action	Steps
Add a new entry	Enter the desired value in the **Resource URIs** field, and then click **Add**.
Modify an existing entry	Use the **Edit**, **Update**, and **Cancel** buttons.
Remove an existing entry	Use the **Delete** and **Undelete** buttons.

Defining access control

On the **Access Control** tab, you can restrict which OAuth clients are allowed to use this access token management instance.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Management** and select your ATM instance or click **Create New Instance**.
2. On the **Access Control** tab, select the **Restrict Allowed Clients** check box.
3. Select a client from the **Allowed Clients** list, and then click **Add**.

   Repeat this step to select additional clients as needed.

Result

To remove a client from the **Allowed Clients** list or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

To disable access control by clients altogether, clear the **Restrict Allowed Clients** check box.

Reviewing the access token management configuration

On the **Summary** tab, review your access token management configuration.

Steps

1.

2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
    - To keep your changes, click **Save**.
    - To discard your changes, click **Cancel**.

Managing access token mappings

In this required configuration, map attributes to be requested from the OAuth resource server into the access token and the token attribute contract.

About this task

When mapping a default context, define how PingFederate maps values into the attributes based on the persistent-grant `USER_KEY`, and any extended attributes defined in **System > OAuth Settings > Authorization Server Settings**. PingFederate acts as an OAuth authorization server.

When a specific context is selected, you can map attributes from the selected context, specifically the chosen IdP adapter instance, Password Credential Validator instance, or authentication policy contract, into the access tokens. You can also map attributes from an IdP connection with an OAuth attribute mapping configuration or an authentication policy contract mapping configuration. You can configure a mapping for clients using the client credential grant type.

The mapping used at runtime depends on the authentication context of the original grant. If the authentication context results in a match, PingFederate uses that specific mapping. Otherwise, it uses the default mapping for the applicable access token manager instance.

> **Note**
>
> The **Access Token Mapping** window becomes available after at least one access token manager (ATM) instance has been configured in **Applications > OAuth > Access Token Management**.

Steps

1. Go to **Applications > OAuth > Access Token Management**.

Action	Steps
Create a mapping	Select the source of the attributes from the **Context** list and the target ATM instance from the **Access Token Manager** list, and then click **Add Mapping**.
Modify an existing mapping	Select it by its name under **Mappings**.
Remove an existing mapping or to cancel the removal request	Click **Delete** or **Undelete** under **Action**. **Note** Before removing an existing mapping from your configuration, ensure that it is not used by your OAuth use cases.

Related links

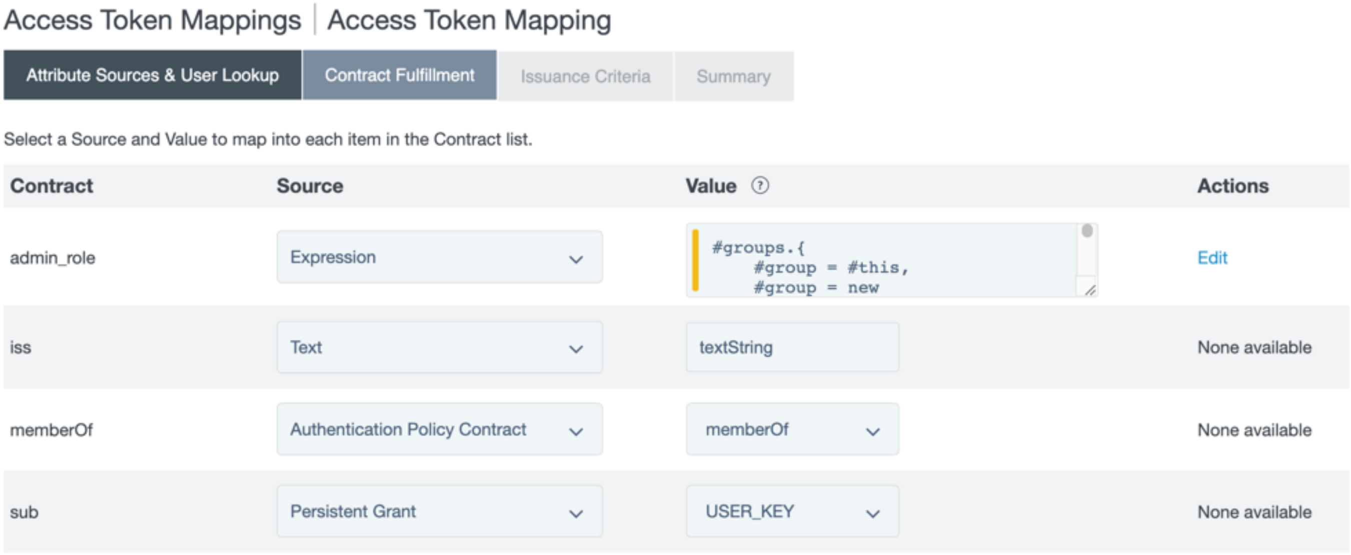
- Mapping OAuth attributes

Configuring access token mapping

Map your policy contract context to the JWT access token manager.

Steps

1. Go go **Applications > OAuth > Access Token Mappings**.
2. On the **Access Token Mappings** page in the **Context** menu, select your policy contract.
3. In the **Access Token Manager** menu, select your JWT ATM.
4. Click **Add Mapping**.
5. On the **Attribute Sources & User Lookup** tab, click **Next**.
6. On the **Contract Fulfillment** tab, select a **Source** and a **Value** to map into the `admin_role`, `iss`, `memberOf`, and `sub` attributes in the **Contract** list.


Access Token Mappings | Access Token Mapping
Attribute Sources & User Lookup
Contract Fulfillment
Issuance Criteria
Summary

Select a Source and Value to map into each item in the Contract list.

Contract	Source	Value	Actions
admin_role	Expression	`#groups.{ #group = #this, #group = new`	Edit
iss	Text	textString	None available
memberOf	Authentication Policy Contract	memberOf	None available
sub	Persistent Grant	USER_KEY	None available

   1. For the `admin_role` attribute, select **Expression** in the **Source** menu and, in the **Value** field, enter the following expression:

```
#filter1 = "^pf_admins.",
#filter2 = "^pf_cryptoadmins.",
#filter3 = "^pf_useradmins.*",
#role1 = "admin",
#role2 = "cryptoadmin",
#role3 = "useradmin",
#role4 = "expressionadmin",

#outboundattribute = new java.util.ArrayList(),

#groups = #this.get("apc.memberOf")!=null?#this.get("apc.memberOf").getValues():{},

#i = 0,

#groups.{
#group = #this,
#group = new javax.naming.ldap.LdapName(#groups[#i]),
#cn = #group.getRdn(#group.size() - 1).getValue().toString(),

#cn.matches(#filter1)?#outboundattribute.add(#role1):null,
#cn.matches(#filter1)?#outboundattribute.add(#role4):null,
#cn.matches(#filter2)?#outboundattribute.add(#role2):null,
#cn.matches(#filter3)?#outboundattribute.add(#role3):null,

#i = #i + 1},

#outboundattribute.size() > 0 ? new
org.sourceid.saml20.adapter.attribute.AttributeValue(#outboundattribute):null
```

> **Note**
>
> This example OGNL expression gets the `memberOf` value from the policy contract, looks for group distinguished name (DN) that match the filters, and assigns a role when a filter is matched. In the expression, anyone that is in the Admins group is assigned both the Admin and Expression Admin role, because the Expression Admin role requires the Admin role assignment. Using this expression to map roles allows you to control access with groups from your identity provider's data source. Match your filter values in the expression to the group names created in your LDAP directory to assign those roles.

2. For the `iss` attribute, select **Text** in the **Source** menu, and enter a text string in the **Value** field.

> **Note**
>
> Make a note of the text string. The value entered here is the issuer claim value and should identify the organization as the issuer.

3. For the `memberOf` attribute, select **Authentication Policy Contract** in the **Source** menu, and **memberOf** in the **Value** menu.

4. For the `sub` attribute, select **Persistent Grant** in the **Source** menu, and **USER_KEY** in the **Value** menu.

5. Click **Next**.

7. On the **Issuance Criteria** tab, click **Next**.

8. On the **Summary** tab, review your mappings. Click **Save**.

Configuring access token attribute sources and user lookup

You can optionally set up datastore queries to supplement values returned from the access token attribute source.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Mapping** and select your mapping, or click **Add Mapping**.

   Choose from:

   - To set up datastore queries, click **Add Attribute Source** and complete the task in the **Attribute Sources & User Lookup** tab. For configuration steps, see Datastore query configuration.
   - To skip this optional configuration, click **Next**.

Configuring access token fulfillment

On the **Contract Fulfillment** tab, map values into the token attribute contract to be included or referenced in the access token.

Steps

1. Choose a source from the **Source** list, and then select a value from the **Value** list for each attribute in the contract, or enter your own.

   Map each attribute from one of the following sources:

   - **Client Credentials**, **IdP Adapter**, **IdP Connection**, **Password Credential Validator**, or **Token Exchange Processor Policy**

     Depending on the selections under **Context** in the **Access Token Attribute Mapping** tab, you can map attributes from that specific authentication system. Select the corresponding context under **Source** and the desired attribute under **Value**.
   - **Persistent Grant**

     When selected, the associated **Value** list is populated with the `USER_KEY` and extended attributes from the persistent access-token grant.
   - **Context**

     Values are returned from the context of the transaction at runtime.

> **Note**
>
> The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are preferred to evaluate and return values.
> Select **Expression** under **Source**, and then click **Edit** to enter an expression.
> The **HTTP Request** Java object retrieves the authentication method that a client uses, or the private key JWT for client authentication if the client uses the private_key_jwt authentication method. For sample expressions, see Expressions for OAuth and OpenID Connect uses cases.
> If the **Expression** selection is not available, you can enable it by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

- **Extended Client Metadata**

  Values are returned from the client record.

- **LDAP/JDBC/Other**

  Values are returned from your datastore, if used. When you make this selection, the Value list populates with attributes from the datastore.

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

- **No Mapping**

  This option ignores the **Value** field, causing no value selection to be necessary.

- **Text**

  The value is what you enter. This can be text only, or you can mix text with references to the `USER_KEY` using the `${USER_KEY}` syntax.

When applicable, you can also enter values from your datastore using the `$\{ds.attribute}` syntax, where `attribute` is any of the datastore attributes you have selected.

+

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Next**.

Defining issuance criteria for access token mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Mapping** and select your mapping, or click **Add Mapping**.
2. On the **Issuance Criteria** tab, select the attribute's source from the **Source** list.
3. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

Source	Description
Mapped from Context (Adapter, Authentication Policy Contract, IdP Connection, Password Credential Validator, token exchange Processor Policy)	Select to evaluate attributes from the authentication source. Visible and applicable only when configuring an access token mapping where the source of the attribute is something other than **Client Credentials** and **Default**. See Managing access token mappings.
Persistent Grant	Select to evaluate the default attribute `USER_KEY` and other extended attributes (if defined) from the persistent grant. Visible and applicable only when configuring an access token mapping where the source of the attribute is not **Client Credentials**.

4. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.
2. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing the access token mapping

On the **Summary** tab, review your access token mapping configuration.

Steps

1. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Configuring an OAuth assertion grant IdP connection

An OAuth assertion grant connection exchanges a SAML assertion or a JSON web token (JWT) for an OAuth access token with the PingFederate OAuth authorization server.

About this task

You can configure an OAuth assertion grant connection with an identity provider (IdP) partner either in conjunction with browser-based single sign-on (SSO), WS-Trust, or independently.

For more information, see Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants and JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants.

Steps

1. Go to **Authentication** > **Integration** > **IdP Connections** and then click **Create Connection**.
2. On the **Connection Type** tab, select the **OAuth Assertion Grant** check box.

> **Tip**
>
> You can also select other options, such as the **Browser SSO Profiles** check box. If you do, you will be prompted to complete the required configuration. This topic only focuses on the **OAuth Assertion Grant** configuration.

3. On the **General Info** tab, enter the required information.
4. On the **OAuth Assertion Grant Attribute Mapping** tab, click **Configure OAuth Assertion Grant Attribute Mapping**.

Defining an attribute contract for the OAuth assertion grant

About this task

An attribute contract is a set of user attributes the IdP sends in the SAML assertions or JWTs for this connection. You identity these attributes on the **OAuth Assertion Grant Attribute Mapping** > **Attribute Contract** window.

`TOKEN_SUBJECT` represents the name identifier of the user for whom the access token is being requested, the `SAML_SUBJECT` attribute in SAML assertions and the `sub` claim in JWTs.

Optionally, you can mask the values of attributes (other than `TOKEN_SUBJECT`) in the log files that PingFederate writes when it receives security tokens.

Steps

- To add an attribute, follow these steps:
    1. Enter the attribute name in the text box.

       Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.
    2. Select the check box under **Mask Values in Log**.
    3. Click **Add**.

- To modify an attribute name or masking selection, follow these steps:
    1. Click **Edit** under **Action** for the attribute.
    2. Make the change and click **Update**.

    > **Note**
    >
    > If you change your mind, ensure that you click **Cancel** under **Action**.

- To delete an attribute, click **Delete** under **Action** for the attribute.

Configuring access token manager mappings

To define how access tokens are created, use the **Access Token Manager Mapping** tab to associate one or more access token manager instances with this connection .

Steps

1. Within your IdP connection configuration, go to the **Access Token Manager Mapping** tab within the **OAuth Assertion Grant Attribute Mapping** tab.

2. Perform one of the following actions.

Action	Steps
Create a new Access Token Manager mapping configuration	Click **Create New Access Token Manager Mapping**.
Edit an existing Access Token Manager mapping configuration	Click the mapping configuration's name.
Delete an Access Token Manager mapping	Click **Delete** under **Action** for the applicable mapping configuration.

Selecting an access token manager instance

This configuration maps attribute values from the (identity provider) IdP connection into the access token to define the resulting content of the access token.

Steps

1. Within your IdP connection configuration, go to the **Access Token Manager** tab.
    1. Click the **OAuth Assertion Grant Attribute Mapping** tab, and then click **Configure OAuth Assertion Grant Attribute Mapping**.
    2. Click the **Access Token Manager Mapping** tab, and then click **Create New Access Token Manager Mapping**.

2. From the **Access Token Manager** list, select an access token manager instance.

> **Tip**
>
> If the access token manager instance you need is not available, click **Manage Access Token Management Instances** to define one or more instances for this connection.

Configuring a datastore for OAuth assertion grant attribute mapping

You can optionally set up datastore queries to supplement values returned from the OAuth assertion grant attribute mapping source.

Steps

1. Within your identity provider (IdP) connection configuration, go to the **Data Store** tab.
   1. Click the **OAuth Assertion Grant Attribute Mapping** tab, and then click **Configure OAuth Assertion Grant Attribute Mapping**.
   2. Click the **Access Token Manager Mapping** tab, and then click **Create New Access Token Manager Mapping**.
   3. Click the **Data Store** tab.
2. Perform one of the following actions.

   Choose from:

   - To set up datastore queries, select a datastore from the **Active Data Store** list, and then click **Next**. Follow the wizard to complete the setup. For configuration steps, see Datastore query configuration.
   - To skip this optional configuration, select **No Data Store** and then click **Next**.

Configuring OAuth assertion grant contract fulfillment

Map values from the SAML assertions or JSON web tokens (JWTs) to the attributes defined for the attribute contract. The access token manager instance requires these values to create an OAuth access token.

About this task

At runtime, a single sign-on (SSO) operation fails if PingFederate cannot fulfill the required attribute.

Steps

1. On **OAuth Assertion Grant Attribute Mapping > OAuth Assertion Grant Attribute Mapping Configuration > Contract Fulfillment** , select a source from the **Source** list and then choose or enter a value for each attribute.
   - **Assertion**

When selected, the **Value** list populates with attributes from the SAML assertion or the JWT.

For example, to map the value of `SAML_SUBJECT` from a SAML assertion, or `sub` from a JWT, as the value of an attribute on the access-token contract, select **Assertion** from the **Source** list and **TOKEN_SUBJECT** from the **Value** list.

- **Context**

  When selected, the **Value** list populates with the available context of the transaction.

  > **Note**
  >
  > Because the **HTTP Request** context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values. For more information, see **Expression**.

- **Extended Client Metadata**

  Values are returned from the client record.

- **LDAP**, **JDBC**, or **Other**

  When selected, the **Value** list is populated with attributes that you have selected from the datastore. Select the desired attribute from the list.

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

  Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  Select this option to ignore the **Value** field.

- **Text**

  When selected, the text you enter is used at runtime. You can mix text with references to any of the values from the SSO token, using the `$\{attribute}` syntax.

  When applicable, you can enter values from your datastore using the `$\{ds.<attribute>}` syntax, where `<attribute>` is any attribute that you have selected from the datastore.

  > **Tip**
  >
  > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Next**.

Defining issuance criteria for OAuth assertion grants

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Within your identity provider (IdP) connection configuration, go to **OAuth Assertion Grant Attribute Mapping > OAuth Assertion Grant Attribute Mapping Configuration > Issuance Criteria**.

2. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Assertion	Select to evaluate attributes from the IdP connection.
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Properties	Select to evaluate OAuth client metadata.

Source	Description
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

3. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.
2. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing OAuth assertion grant attribute mapping configuration

On the **Summary** tab, review your OAuth assertion grant attribute mapping configuration.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Reviewing OAuth assertion grant configuration

On the **Summary** tab, review your OAuth assertion grant configuration.

Steps

1.
   - ◦ To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - ◦ To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - ◦ To discard your changes, click **Cancel**.

Configuring OpenID Connect policies

This configuration allows you to define OpenID Connect policies for client access to attributes mapped according to OpenID specifications.

To begin configuring OpenID Connect policies, go to **Applications** > **OAuth** > **OpenId Connect Policy Management**.

On the **OpenID Connect Policy Management** page, you can do the following:

- Configure a new OpenID Connect policy by clicking **Add Policy**.
- Modify an existing OpenID Connect policy by selecting it's name under **Policy ID**.
- Review the usage of an existing OpenID Connect policy by clicking **Check Usage** under **Action**.
- Remove an existing OpenID Connect policy or cancel the remove request by clicking **Delete** or **Undelete** under **Action**.
- Chose an existing OpenID Connect policy to be the default policy by clicking **Set as Default** under **Action**.

Configuring policy and ID token settings

Configure your OpenID Connect policy settings and the required and optional information for ID tokens.

Steps

1. Go to **Applications** > **OAuth** > **OpenID Connect Policy Management** and click **Add Policy**.
2. In the **Policy ID** field, enter the policy identifier.
3. In the **Name** field, enter the policy name.
4. In the **Access Token Manager** list, select an access token management instance.
5. **Optional:** In minutes, define the expiry information for ID tokens issued based on this policy in the **ID Token Lifetime** field.

   The default value is `5` minutes.

6. **Optional:** Select the **Include Session Identifier in ID Token** check box to add a session identifier (`pi.sri`) in the ID tokens.

   Doing this might be useful for the relying parties, such as PingAccess, for client session management.

7. **Optional:** Select the **Include User Info in ID Token** check box to include additional attributes in the ID tokens.

   > **Tip**
   >
   > OAuth clients can also obtain additional attributes from the UserInfo endpoint at `/idp/userinfo.openid` . For more information, see UserInfo endpoint.

8. **Optional:** Select the **Include State Hash in ID Token** check box to include the `s_hash` claim in ID tokens.

   > **Note**
   >
   > A state hash protects the state parameter by binding it to the ID token. For more information, see Financial Services – Financial API - Part 2: Read and Write API Security Profile.

9. **Optional:** Select the **Include X.509 Thumbprint Header in ID Token** check box to include the `x5t` header parameter for the token.

   > **Note**
   >
   > The X.509 thumbprint (`x5t`) is only included in the `ID Token` header when static keys are enabled. For more information, see Configuring static signing keys.

10. **Optional:** In the **ID Token Type (TYP) Header Value** field, enter the token type. This field indicates the value of the Type (`typ`) header in the JSON Web Token (JWT). If you do not specify a header, it is omitted.

    > **Tip**
    >
    > Use `JWT` in the **ID Token Type (TYP) Header Value** field to indicate that the object is a JWT. For compatibility with older implementations, it's best to always spell `JWT` in uppercase, even though media type names are not case-sensitive.

11. **Optional:** Select the **Return ID Token On Refresh Grant** check box to return an ID token for OpenID Connect to Salesforce and Kubernetes when the OAuth access token is refreshed.

12. **Optional:** Select the **Reissue ID Token In Hybrid Flow** check box to issue a new ID token at the token endpoint that is different from the first ID token issued for an authorization endpoint request.

    This is applicable only for OpenID Connect hybrid flows. For more information about hybrid flows, see .net/specs/openid-connect-basic-1_0.html//[Protocol Elements] in the OpenID Connect Basic Client Implementer's Guide.

    > **Tip**
    >
    > To modify the personally identifiable information (PII) in the ID token, see Configuring ID token fulfillment.

Configuring the policy attribute contract

In the **Attribute Contract** tab, you can define the list of attributes that PingFederate can return to the OAuth clients.

About this task

Every new OpenID Connect policy contract begins with a list of standard attributes. These attributes or claims are defined in the OpenID Connect specification. You can optionally remove standard attributes, turn them into non-standard attributes, or add new non-standard attributes.

> **Note**
>
> In OpenID Connect, scopes affect the list of attributes that PingFederate can return to the OAuth clients. The attributes that PingFederate returns to OAuth clients vary, depending on the scopes originally approved by the resource owner.

By default, all attributes defined on this window are deliverable through the UserInfo endpoint. If an implicit client makes a token request by providing `id_token` as the only `response_type` parameter value, the client will only receive an ID token without an access token. As the client will not be able to retrieve additional attributes from the UserInfo endpoint without a valid access token, PingFederate includes the applicable attributes in the ID token instead.

If you have not selected the **Include User Info in ID Token** option in the **Manage Policy** tab for this policy, you can choose how attributes are delivered to clients. Similar to the default delivery behavior, in the scenario where an implicit client makes a token request by providing `id_token` as the sole `response_type` parameter value, PingFederate includes the applicable attributes in the ID token regardless of any configured overrides.

Steps

- To add a new attribute:
  1. Enter the name of the attribute under **Extend the Contract**.
  2. **Optional:** To choose how the attribute is delivered, select the **Override Default Delivery** check box .
     - To include this attribute in ID tokens, select the **ID Token** check box.
     - To include this attribute in Userinfo responses, select the **UserInfo** check box.
  3. **Optional:** To always return this array in a token response, select the **Multi-Valued** check box.
  4. Click **Add**.
- To modify an existing entry, use the **Edit**, **Update**, and **Cancel** buttons. Choose how the attribute is delivered, as needed.
- To remove an existing entry, click **Delete**.

Related links

- UserInfo endpoint

Configuring attribute scopes

With OpenID Connect, scopes affect the list of attributes that PingFederate can return to the OAuth clients. In the **Attribute Scopes** tab, you can optionally add associations between scopes and attributes beyond what is defined in the specification.

Steps

1. Go to **Applications** > **OAuth** > **OpenID Connect Policy Management** and select your policy, or click **Add Policy**.

2. In the **Attribute Scopes** tab, add any number of scope-to-attributes associations.

    1. Select a scope from the **Scope** list.

        Common and exclusive scopes are both available.

    2. Select the relevant check boxes under **Attributes**.

        > **Note**
        >
        > If you have selected a standard scope, its associated standard attributes are automatically selected and cannot be modified. You can select additional attributes to be associated with the selected scope.
        > If you have selected the `profile` scope, any non-standard attributes that are not associated with the `profile` scope become inaccessible to your OAuth clients. The administrative console displays a warning message with a list of inaccessible attributes. Select the relevant check boxes to make the non-standard attributes accessible, or ignore the message to leave them inaccessible for now.

    3. Click **Add**.

    4. **Optional:** Repeat these steps to define additional scope-to-attributes associations.

    Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

3. Click **Next**.

Related links

- Requesting Claims using Scope Values, from the OpenID Connect specification
- Defining scopes

Configuring policy attribute sources and user lookup

You can optionally set up datastore queries to supplement values returned from the policy attribute source.

Steps

1. Go to **Applications** > **OAuth** > **OpenID Connect Policy Management** and select your policy, or click **Add New Policy**.

2. On the **Attribute Sources & User Lookup** tab, perform one of the following actions.

    Choose from:

    - To set up datastore queries, click **Add Attribute Source**. Complete the setup on the **Attribute Sources & User Lookup** tab. For configuration steps, see Datastore query configuration.
    - To skip this optional configuration, click **Next**.

Configuring ID token fulfillment

Map attributes from the access token or other sources to fulfill the attribute contract.

Steps

1. Go to **Applications** > **OAuth** > **OpenID Connect Policy Management** and select your policy, or click **Add Policy**.
2. On the **Contract Fulfillment** tab, select a source from the **Source** list and then select or enter a value for each attribute in the contract.

   Map the subject attribute and all extended attributes from one of the following sources:

   ### *Context*

   Values are returned from the context of the transaction at runtime.To enter an expression, select **Expression** under **Source**, and then click **Edit**.

   > **Note**
   >
   > When modifying the personally identifiable information (PII) for hybrid flows, if the **RequestEndpoint** context value ends with a token endpoint path the actual value is populated and sent in the token response. If the field is blank, a null value is sent in the token response.
   > Because the **HTTP Request** context value is retrieved as a Java object rather than text, OGNL expressions are preferred to evaluate and return values.
   > If **Expression** is not available, you can enable it by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   ### *Extended Client Metadata*

   Values are returned from the client record.

   ### *LDAP/JDBC/Other*

   Values are returned from your datastore, if used. When selected, the **Value** list populates with attributes from the datastore.

   ### *Expression*

   When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

   ### *No Mapping*

   This option ignores the **Value** field.

   ### *Text*

   The value is what you enter. This can be text only, or you can mix text with references to the unique user ID returned from the credentials validator, using the `${attribute}` syntax.You can also enter values from your datastore, when applicable, using the `$\{ds.attribute}` syntax, where `attribute` is any of the datastore attributes you have selected.

   > **Tip**
   >
   > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

Access Token

The value is provided from the access token.

Persistent Grant

Enables direct mapping from the grant to the ID Token and to user information attributes.

3. Click **Next**.

Defining issuance criteria for policy mapping

Individual attributes within policy contracts can further determine whether PingFederate approves or rejects requests. You can define those criteria to satisfy or you can choose to skip this configuration.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. Go to **Applications > OAuth > OpenID Connect Policy Management** and select your policy, or click **Add Policy**.

2. On the **Issuance Criteria** tab, select the attribute's source from the **Source** list.

3. In the **Source** list, select the attribute's source.

   Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Access Token	Select to evaluate attributes from the access token.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The HTTP Request context value is retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
Extended Client Metadata	Select to evaluate OAuth client metadata.
JDBC, LDAP, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Persistent Grant	Select to evaluate attributes from the persistent grant.

4. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.
2. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

   For more information, see Attribute mapping expressions.

   1. Click **Show Advanced Criteria**.
   2. In the **Expression** field, enter the required expressions.
   3. In the **Error Result** field, enter an error code or message.

      > **Note**
      >
      > If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

   4. Click **Add**.
   5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
   6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links

Reviewing your OpenID Connect policy

On the **Summary** tab, review your OpenID Connect policy.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Result

Note

If this is the first policy you are creating, you must click **Done** and designate this policy as the default before saving on the **OpenID Connect Policy Management** window. You can change the default when you create additional policies.

Client Initiated Backchannel Authentication (CIBA)

Client Initiated Backchannel Authentication is an extension to OpenID Connect that improves the end-user experience during authentication and authorization in a federated environment.

The CIBA extension defines a new OAuth grant type where user consent can be requested through an out-of-band flow. CIBA improves the user experience, such as when making an online purchase from a merchant, because it does not require a browser redirect to a financial institution to authorize the purchase. Instead, the user can receive a push notification sent to the financial institution's native mobile app running on the user's phone to complete the authorization. For more information, see openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html.

Note

The PingOne MFA Integration Kit includes the PingOne MFA CIBA Authenticator, which works with PingFederate's CIBA feature. For instructions on configuring the PingOne MFA CIBA Authenticator, see Configuring a CIBA authenticator instance.

A CIBA configuration consists of two components: a CIBA authenticator and a CIBA request policy.

CIBA authenticator

A CIBA authenticator is responsible for authenticating users through an out-of-band method.

You can use the PingFederate SDK to implement a custom solution. For more information, see the Javadoc for the `OOBAuthPlugin` interface, the `SampleEmailAuthPlugin.java` file for a sample implementation, and the SDK developer's guide for build and deployment information.

Once deployed, you can create one or more instance configurations of the authenticator.

For more information, see Configuring a CIBA authenticator instance.

CIBA request policy

CIBA request policies process identity hints and authenticate users to receive consent. Each request policy is associated with an instance of a CIBA authenticator. The CIBA grant flow is initiated by a direct request from the client and involves an out-of-band interaction with the user to complete authentication and authorization. OAuth clients that support the CIBA grant type can be configured to use a specific CIBA request policy or a default.

For more information, see Defining a request policy.

Note

Because the CIBA extension is an OAuth grant type, to enable CIBA for the client, you must select **CIBA** in the **Allowed Grant Types** setting. Once selected, you can configure more client CIBA-related settings.
For more information, see Configuring OAuth clients.

Managing CIBA authenticators

Manage the Client Initiated Backchannel Authentication (CIBA) authenticators in PingFederate.

About this task

A CIBA authenticator is responsible for authenticating users through an out-of-band method.

You can use the PingFederate SDK to implement a custom solution. For more information, see the Javadoc for the `OOBAuthPlugin` interface, the `SampleEmailAuthPlugin.java` file for a sample implementation, and the SDK developer's guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

Once deployed, you can create one or more instance configurations of the authenticator.

Steps

- Go to **Authentication** > **OAuth** > **CIBA Authenticators** and create or select the instance you want to manage.
- To manage the CIBA authenticator, choose from the following options.

Option	Description
Configure a new instance	Click **Create New Instance**
Modify an existing instance	Click the name of instance in the **Instance Name** column
View the usage of an existing instance	Click **Check Usage** in the **Action** column on the instance's row
Remove an existing instance	Click **Delete** in the **Action** column on the instance's row

Configuring a CIBA authenticator instance

The PingOne MFA Integration Kit includes the PingOne SDK client initiated backchannel authentication (CIBA) Authenticator, which works with PingFederate's CIBA feature.

About this task

> **Note**
>
> For instructions on configuring the CIBA Authenticator for PingOne SDK, see Configuring a CIBA authenticator instance.

Steps

1. Go to **Authentication** > **OAuth** > **CIBA Authenticators** to open the **CIBA Authenticators** window.

2. On the **CIBA Authenticators** window, click **Create New Instance** to start the **Create CIBA Authenticator Instance** configuration workflow.

3. On the **Type** tab, configure the basics of this authenticator instance.

    1. Enter a name and an ID in the **Instance Name** and **Instance ID** fields.

    2. Select a CIBA authenticator from the **Type** list.

        Selections vary depending on the deployed CIBA authenticators.

        You can use the PingFederate SDK to implement a custom solution. For more information, see the Javadoc for the `OOBAuthPlugin` interface, the `SampleEmailAuthPlugin.java` file for a sample implementation and the SDK developer's guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

4. On the **Instance Configuration** tab, follow the on-screen instructions to configure the authenticator instance.

    Configuration requirements vary depending on the authenticator implementation.

5. On the **Actions** tab, follow the on-screen instructions to test the validity of the authenticator instance configuration or to perform secondary configuration tasks.

    Availability of this tab and actions vary depending on the authenticator implementation.

6. On the **Extended Contracts** tab, follow the on-screen instructions to define additional attributes.

    The authenticator contract is the list of input parameters used to challenge the user for authentication. Some authenticators support extending the contract for additional functionality, such as formatting the data presented to the user during the authentication challenge.

    Availability of this window and supported attribute names vary depending on the authenticator implementation.

7. On the **Summary** tab, review your configuration, modify as needed, and click **Done** to exit the **Create CIBA Authenticator Instance** workflow.

8. On the **CIBA Authenticators** window, click **Save** to retain the configuration of the authenticator instance.

    If you want to exit without saving the configuration, click **Cancel**.

Managing CIBA request policies

You can configure, modify, review, remove, and elect existing client initiated backchannel authentication (CIBA) request policies in the administrative console.

About this task

CIBA request policies process identity hints and authenticate users to receive consent. Each request policy is associated with an instance of a CIBA authenticator. The CIBA grant flow is initiated by a direct request from the client and involves an out-of-band interaction with the user to complete authentication and authorization. OAuth clients that support the CIBA grant type can be configured to use a specific CIBA request policy or a default.

Steps

1. Go to **Applications** > **OAuth** > **CIBA Request Policies**. To configure a new CIBA request policy, click **Add Policy**.
2. Select an action from the following options:

   Choose from:

   - To modify an existing CIBA request policy, select it by its name under **Policy ID**.
   - To review the usage of an existing CIBA request policy, click **Check Usage** under **Action**.
   - To remove an existing CIBA request policy or to cancel the removal request, click **Delete** or **Undelete** under **Action**.
   - To elect an existing CIBA request policy to be the default CIBA request policy, click **Set as Default** under **Action**.

Defining a request policy

You can define the basics of your client-initiated backchannel authentication (CIBA) request policy in the PingFederate administrative console.

Steps

1. Go to **Applications** > **OAuth** > **CIBA Request Policies**.
2. On the **Manage Policy** tab, define the basics of your CIBA request policy.

   For more information about each field, refer to the following table.

Field	Description
Policy ID (Required)	The unique identifier of this request policy.
Name (Required)	The name of this request policy.
Authenticator (Required)	The CIBA authenticator instance associated with this request policy.
User Code PCV	The Password Credential Validator (PCV) instance that PingFederate uses to validate the `user_code` parameter values it receives from clients associated with this request policy. **Important** If a client is associated with a request policy that has been configured with a PCV instance, it can support user code in its configuration. A client supporting user code must not be associated with a request policy that is not configured with a PCV instance. For more information on CIBA client configuration, see Configuring OAuth clients.

Field	Description
Transaction Lifetime (Seconds)	The validity, in seconds, of authentication requests PingFederate receives from clients associated with this request policy since the generation of their authentication request acknowledgments. The default value is `120`. Clients can request a shorter lifetime by including the `requested_expiry` request parameter in their authentication requests.
Allow Unsigned Login Hint Token	Controls whether clients associated with this request policy can use unsigned JSON web tokens (JWT) as values of the `login_hint_token` request parameter in their authentication requests. This check box is not selected by default.
Require Token for Identity Hint	Controls whether clients associated with this request policy must use either the `id_token_hint` or `login_hint_token` as the identity hint in their authentication requests. This check box is not selected by default. When selected, clients associated with this request policy cannot use `login_hint` as the identity hint in their authentication requests.
Alternative Login Hint Token Issuers	Alternative issuers that clients associated with this request policy can use in their signed login hint tokens. Furthermore, each additional issuer requires either the JWKS url or the actual JWKS so that PingFederate can verify the authenticity of the signed login hint tokens.

3. Click **Next**.

Related links

- Client-initiated backchannel authentication endpoint

Configuring identity hint contract

You can configure the identity hint contract, which contains the set of attributes received in the client initiated backchannel authentication (CIBA) request that identifies the user.

About this task

`IDENTITY_HINT_SUBJECT` is a core attribute and is automatically populated by the `sub` attribute of an identity hint token, if found, or the attribute value of the `login_hint` request attribute.

A client can send an ID token, `id_token_hint`, or a login hint token, `login_hint_token`, as the identity hint token. If you extend the identity hint contract with attribute names from the identity token, PingFederate fulfills them with values found in the identity token.

> **Tip**
>
> As needed, all attributes can optionally be fulfilled differently on the **Identity Hint Contract Fulfillment** tab.

Steps

1. **Optional:** Go to **Applications** > **OAuth** > **CIBA Request Policies**. On the **Identity Hint Contract Fulfillment** tab, enter an attribute name under **Extend the Contract**, and then click **Add**.
2. Repeat the previous step to define additional attributes. Click **Next**.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.

Example

Example

Suppose the following JSON web token (JWT) matches the expected structure of the login hint tokens.

```
{
  "sub": "asmith",
  "attrs": {
    "mail": "asmith@example.com",
    "phone": "555-555-5555"
  }
}
```

To add both the `mail` and `phone` attributes, extend the contract with `login_hint_token.attrs.mail` and `login_hint_token.attrs.phone`, respectively.

Related links

- Client-initiated backchannel authentication endpoint

Configuring identity hint contract fulfillment

You can process the identity hint to further augment the identity data prior to contract fulfillment in the PingFederate administrative console.

Steps

1. Go to **Applications** > **OAuth** > **CIBA Request Policies** > **Identity Hint Contract Fulfillment**.
2. Click **Manage Fulfillment** to begin the mapping configuration.
3. When the administrative console returns you to the **Identity Hint Contract Fulfillment** tab, click **Next**.

Configuring attribute sources and user lookup

You can configure the attribute sources and user lookup values in the PingFederate administrative console.

About this task

You can set up datastore queries to supplement values returned from the source. This configuration to fulfill the identity hint's attribute contract is optional.

Steps

1. Go to **Applications** > **OAuth** > **CIBA Request Policies**.
2. Choose one of the following actions:

   Choose from:

   - On the **Attribute Sources & User Lookup** tab, to set up datastore queries, click **Add Attribute Source**.
   - To skip this optional configuration, click **Next**.

     Follow the configuration workflow to complete the setup. For configuration steps, see Datastore query configuration.

   When the administrative console returns you to the **Attribute Sources & User Lookup** tab, click **Next**.

Fulfilling identity hint contract

You can fulfill identity hint contracts in PingFederate.

About this task

On the Identity Hint Contract Fulfillment tab, fulfill the identity hint contract with values from the original identity hint, datastores, dynamic text values, or attribute mapping expressions, if enabled.

Steps

1. From the **Source** list, select a source.

   For more information about the **Source** list, see the following table.

Source	Description
Context	Select **Context** to return specific information from the request.
JDBC, LDAP, or other types of datastore (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a datastore. When you make this selection, the list under **Value** populates with attributes from your database, directory, or other datastore. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** tab. For more information, see Configuring attribute sources and user lookup.
Request	Select **Request** to use the attribute value PingFederate found in the CIBA request without customization.

Source	Description
Expression (if enabled)	Select **Expression** to support complex mapping requirements, such as transforming incoming values into different formats. Additionally, HTTP request is retrieved as a Java object rather than text. For this reason, select **Expression** as the source and use OGNL expressions to evaluate and return specific information from the HTTP request. Applicable only if you have enabled the use of expressions in PingFederate. For more information, see Attribute mapping expressions.
No Mapping	Select **No Mapping** to ignore the **Value** field, making value selection unnecessary.
Text	Select **Text** to return the value you enter under **Value**. You might use a static text value if the target web application provides a service based on the name of your organization. You can mix text with references to attributes from the identity provider (IdP) adapter contract by using the `$\{<attribute>}` syntax. `$\{ds.<attr-source-id>.<attribute>}` You can also enter references to syntax, where `<attr-source-id>` is the **Attribute Source ID** value you entered on the **Data Store** tab and `<attribute>` is an attribute from the datastore. **Tip** You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Specify a value associated with the selected source.

   Not applicable if you selected **Request**. You can also enter references to attributes from configured attribute sources by using the **Source** list.

3. Repeat these steps until all attributes are configured.

4. Click **Next**.

Defining issuance criteria for identity hint contract

You must define issuance criteria for an identity hint contract to further process a request in PingFederate.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.
2. Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** context value is retrieved as a Java object instead of text, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, LDAP, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Request	Select to evaluate attributes from the CIBA request.

3. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**

- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.
2. Click **Next**.

Related links

Reviewing identity hint contract fulfillment

You can review, modify, or discard identity hint contract fulfillment configurations in PingFederate.

About this task

On the **Summary** tab, review your configuration.

Steps

1. Perform the following actions as needed.

Action	How to accomplish it
Amend your configuration	Click the corresponding tab and follow the configuration workflow
Keep your changes	Click **Done** and continue with the configuration
Discard your changes	Click **Cancel**

Configuring attribute sources and user lookup for request policy contract

You can configure the attribute sources and user lookup for Client-Initiated Backchannel Authentication (CIBA) request policy contracts in the PingFederate administrative console.

About this task

You can optionally set up datastore queries to supplement values returned from the source.

Steps

1. Go to **Applications** > **OAuth** > **CIBA Request Policies**.
2. Choose one of the following actions:

   Choose from:

   - On the **Attribute Sources & User Lookup** tab, to set up datastore queries, click **Add Attribute Source**.
   - To skip this optional configuration, click **Next**.

     Follow the configuration workflow to complete the setup. For configuration steps, see Datastore query configuration.

   When the administrative console returns you to the **Attribute Sources & User Lookup** tab, click **Next**.

Configuring request policy contract fulfillment

You can fulfill the request policy contract in PingFederate.

About this task

On the **Contract Fulfillment** tab, fulfill the request policy contract with values from the original identity hint, datastores, dynamic text values, or attribute mapping expressions (if enabled).

This contract is used to map into the OAuth grant (the `USER_KEY` attribute), the Client Initiated Back channel Authentication (CIBA) authenticator (attributes vary depending on the authenticator), and the user code Password Credential Validator (PCV) (the `USER_CODE_USER_NAME` attribute). The `USER_CODE_USER_NAME` attribute is shown only if a PCV instance is selected on the **Manage Policy** window.

Steps

1. Select a source from the **Source** list.

   For more information about the **Source** list, see the following table.

Source	Description
Context	Select **Context** to return specific information from the request.
JDBC, LDAP, or other types of datastores (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a datastore. When you make this selection, the list under **Value** is populated with attributes from your database, directory, or other datastore. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** window. For more information, see Configuring attribute sources and user lookup for request policy contract.
Request	Select **Request** to use the attribute value PingFederate found in the CIBA request without customization.
Expression (if enabled)	Supports complex mapping requirements, such as transforming incoming values into different formats. Additionally, the HTTP request is retrieved as a Java object rather than text. Therefore, select **Expression** as the source and use OGNL expressions to evaluate and return specific information from the HTTP request. Applicable only if you have enabled the use of expressions in PingFederate. For more information, see Attribute mapping expressions.
No Mapping	Select **No Mapping** to ignore the **Value** field, making value selection unnecessary.
Text	Select **Text** to return the value you entered under **Value**. You might use a static text value if the target web application provides a service based on the name of your organization. You can provide the attribute value as a constant. You can mix text with references to attributes from the identity provider (IdP) adapter contract by using the `$\{<attribute>}` syntax. You can also enter references to attributes from configured attribute sources by using the `$\{ds.<attr-source-id>.<attribute>}` syntax, where `<attr-source-id>` is the **Attribute Source ID** value you entered on the **Attribute Sources & User Lookup > Data Store** tab and `<attribute>` is an attribute from datastore. **Tip** You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Specify a value associated with the selected source.
3. Repeat these steps until all attributes are configured.
4. Click **Next**.

Defining issuance criteria for CIBA request policy

You must define the issuance criteria for PingFederate to further process a client-initiated backchannel authentication (CIBA) request policy.

About this task

On the **Issuance Criteria** tab, define the criteria to satisfy for PingFederate to further process a request. Use this token authorization feature to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by choosing the source that contains the attribute to verify. Some sources are common to almost all use cases, such as **Mapped Attributes**. Other sources depend on the type of configuration, such as **JDBC**. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to verify. Depending on the selected source, the available attributes or properties vary. Specify the comparison condition and the desired value to compare to.

You can define multiple criteria, which must all be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value, depending on the chosen comparison method. The **multi-value contains ...** or **multi-value does not contain ...** comparison methods are intended for attributes that can contain multiple values. Such a criterion is considered satisfied if one of the multiple values match or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward, regardless of how the criteria were defined. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.
2. Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, LDAP, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.

Source	Description
Mapped Attributes	Select to evaluate the mapped attributes.
Request	Select to evaluate attributes from the CIBA request.

3. In the **Attribute Name** list, select the attribute to be evaluated.

   Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ This value is used by the `error_description` protocol field. Using an error code in the **Error Result** field allows an application to process the code in several ways, such as displaying an error message or e-mailing an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

1. **Optional:** Repeat to add more criteria.

2. Click **Next**.

Related links

Reviewing your CIBA request policy

Client Initiated Backchannel Authentication (CIBA) is an extension to OpenID Connect that can improve the end-user experience during authentication and authorization in a federated environment. You can review the request policy in PingFederate.

Steps

1. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Result

> **Note**
>
> If this is the first policy you are creating, you must click **Done** and designate the first policy as the default before saving on the**CIBA Request Policies** window. You can change the default as needed when you create additional policies.

OAuth attribute mapping using a datastore

Although an optional configuration, you can map OAuth attributes using a datastore in the PingFederate administrative console.

About this task

This optional configuration is the same for all OAuth grant mapping and token mapping configurations.

Steps

1. For grant mapping, go to **Authentication** > **OAuth** > **IdP Adapter Grant Mapping**. For more information, seeGrant contract mapping.

2. For token mapping, go to **Applications** > **OAuth** > **Access Token Mapping**. For more information, see Token mapping.

OAuth client session management

When an organization opens web-based protected resources to their remote employees, business partners, and customers, it has limited control over the end-user devices. To minimize security risk, both the IT administrators and end users desire session management with tight controls.

PingFederate provides an Asynchronous Front-Channel Logout endpoint and a Back-Channel Session Revocation Web Service to help OAuth clients, such as PingAccess, to terminate sessions when end users log out and to prevent unauthorized access until the end users log in again.

PingAccess works out-of-the-box with PingFederate, taking full advantages of these two features.

Asynchronous Front-Channel Logout

Asynchronous Front-Channel Logout provides OAuth clients the capability to initiate single logout (SLO) requests to sign off associated SLO-enabled OpenID Connect (OIDC), SAML 2.0, or WS-Federation sessions.

The Asynchronous Front-Channel Logout endpoint is `/idp/startSLO.ping`. Optionally, clients can add end-user sessions to a revocation list on logout and query the revocation list through the Back-Channel Session Revocation endpoint.

> **Tip**
>
> The Asynchronous Front-Channel Logout endpoint is also published in the OIDC metadata at the `/.well-known/openid-configuration` endpoint. Look for `ping_end_session_endpoint` in the metadata.

You can set the logout mode for a client as Ping Front-Channel or OIDC Back-Channel. Learn more in Configuring OAuth clients.

When you select Ping Front-Channel, PingFederate sends logout requests, using the browser, to PingAccess and additional requests to other relying parties.

When you select the PingAccess option, PingFederate sends logout requests, using the browser, to the OIDC logout endpoint on PingAccess(`/pa/oidc/logout.png`) to sign off other domains previously called by the session. For more information, see OpenID Connect endpoints in the PingAccess documentation.

When you select OIDC Back-Channel, PingFederate sends a logout token to the client's configured Back-Channel Logout URI. This feature conforms to the OpenID Connect Back-Channel Logout specification.

In addition, when signing off an SLO-enabled SAML 2.0 or WS-Federation session, because the service provider (SP)-initiated logout request reaches the PingFederate identity provider (IdP) server, the same logout process applies as well. Depending on the enterprise architecture, this could further improve single sign-on (SSO) and logout use cases.

Back-Channel Session Revocation

Back-Channel Session Revocation allows OAuth clients, such as PingAccess, to query the revocation status of their sessions by sending HTTP GET requests to the session revocation endpoint on PingFederate at `/pf-ws/rest/sessionMgmt/revokedSris`.

To access the session revocation endpoint, a client must be granted access to the Session Revocation API. It must also authenticate with its client secret or client certificate and include in the request the session identifier, which can be obtained from the access token or the ID token.

Back-Channel Session Revocation also allows the clients to revoke sessions by sending HTTP POST requests to the same session revocation endpoint. This gives application developers the flexibility to revoke sessions based on the logic of their applications.

For each session added to the revocation list, PingFederate retains its revocation status for a configurable lifetime. Access control and authentication requirements to revoke sessions are identical to those to query for the revocation status.

OAuth token exchange

By configuring the OAuth authorization server to support OAuth token exchange, the authorization server can exchange a client's security token for another type of token.

The OAuth token exchange allows resource servers to exchange access tokens for other security tokens that are required to call additional APIs, much like what the microservices architecture requires. PingFederate's native support of subject tokens and actor tokens opens new use cases around delegation and impersonation that enrich the end-user experience as resources flow through seamlessly among the back-end services used by the user-facing applications.

PingFederate can perform the following token exchanges:

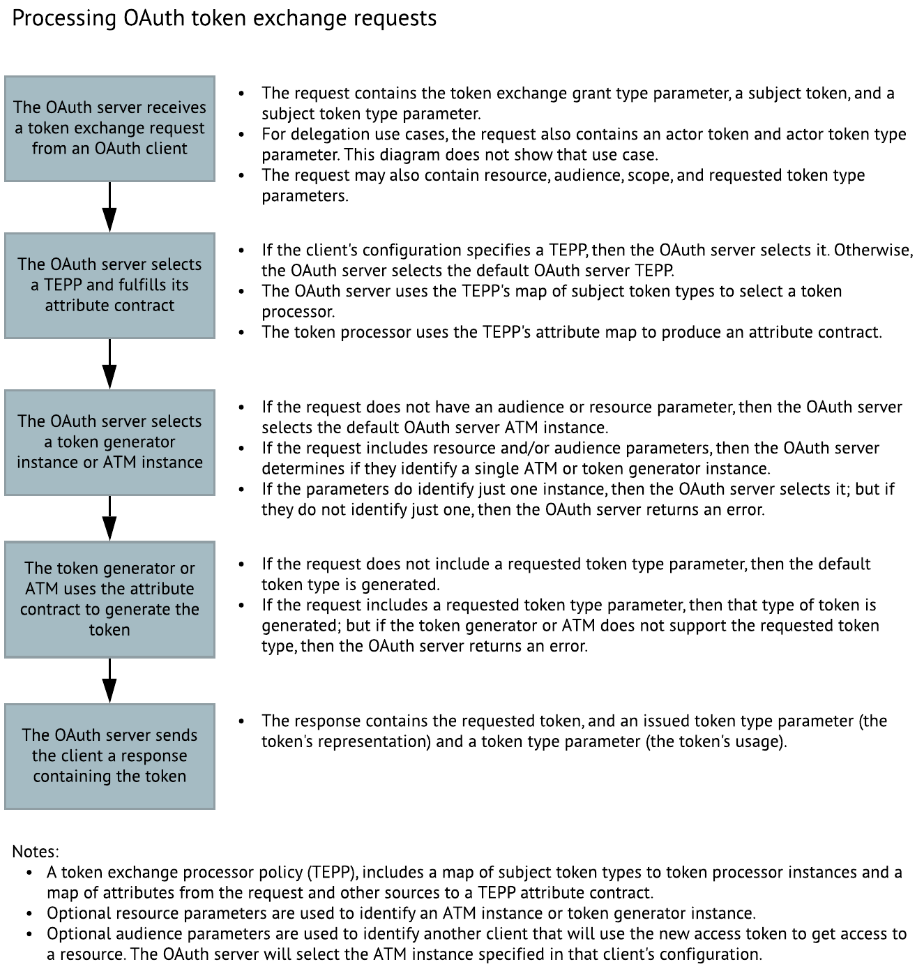
- Access tokens for access tokens
- ID tokens for access tokens
- SAML (1.1, 2.0) tokens for access tokens
- SAML tokens for SAML tokens
- Kerberos tokens for access tokens
- Kerberos tokens for SAML tokens
- X.509 certificate tokens for access tokens
- X.509 certificate tokens for SAML tokens

An OAuth token exchange transaction begins when an OAuth client sends the authorization server a request with the token exchange grant type. Then the authorization server gets the token exchange processor policy specified in the client's configuration.

The token exchange processor policy specifies which parameters from the token exchange request, and optionally which attributes from other sources, will be used. The policy always uses the subject token, but it can use an actor token too. The policy also specifies which token processor instance to use based on the request's subject token type and actor token type when present. During the transaction, the token processor instance transforms the subject token and optionally the actor token, parameters, and attributes into a token exchange processor policy attribute contract.

Depending on the type of token requested and what will consume it, the authorization server sends the attribute contract to a token generator instance or access token manager instance to generate the requested token.

This feature uses the protocol defined in the specification for OAuth 2.0 Token Exchange.



Processing OAuth token exchange requests

The OAuth server receives a token exchange request from an OAuth client
The OAuth server selects a TEPP and fulfills its attribute contract
The OAuth server selects a token generator instance or ATM instance
The token generator or ATM uses the attribute contract to generate the token
The OAuth server sends the client a response containing the token

- The request contains the token exchange grant type parameter, a subject token, and a subject token type parameter.
- For delegation use cases, the request also contains an actor token and actor token type parameter. This diagram does not show that use case.
- The request may also contain resource, audience, scope, and requested token type parameters.

- If the client's configuration specifies a TEPP, then the OAuth server selects it. Otherwise, the OAuth server selects the default OAuth server TEPP.
- The OAuth server uses the TEPP's map of subject token types to select a token processor.
- The token processor uses the TEPP's attribute map to produce an attribute contract.

- If the request does not have an audience or resource parameter, then the OAuth server selects the default OAuth server ATM instance.
- If the request includes resource and/or audience parameters, then the OAuth server determines if they identify a single ATM or token generator instance.
- If the parameters do identify just one instance, then the OAuth server selects it; but if they do not identify just one, then the OAuth server returns an error.

- If the request does not include a requested token type parameter, then the default token type is generated.
- If the request includes a requested token type parameter, then that type of token is generated; but if the token generator or ATM does not support the requested token type, then the OAuth server returns an error.

- The response contains the requested token, and an issued token type parameter (the token's representation) and a token type parameter (the token's usage).

Notes:

- A token exchange processor policy (TEPP), includes a map of subject token types to token processor instances and a map of attributes from the request and other sources to a TEPP attribute contract.
- Optional resource parameters are used to identify an ATM instance or token generator instance.
- Optional audience parameters are used to identify another client that will use the new access token to get access to a resource. The OAuth server will select the ATM instance specified in that client's configuration.

Configuring OAuth token exchange

Configuring the OAuth authorization server to support OAuth token exchange involves configuring token exchange processor policies, token generator instances and token exchange generator groups, access token manager instances, and OAuth clients.

About this task

To configure OAuth token exchange, see the included topic links to perform the necessary steps.

> **Tip**
>
> Temporary AWS security credentials are security token service (STS) tokens. To exchange inbound STS tokens, use PingFederate's SAML 2.0 token processor and the configured SAML 2.0 token processor policy in the token exchange processor policy instance. The details depend on your requirements.

Steps

1. Define token exchange processor policies to handle incoming token exchange requests. See Defining token exchange processor policies.
2. If you need token generator instances to generate the requested tokens, complete the following tasks.
    1. Configure the token generator instances. See Managing token generators.

    2. Create token exchange generator groups. See Creating token exchange generator groups.

    3. Map the attributes from the token exchange processor policies to the attributes from the token generator instances. See Mapping token exchange attributes to token generator attributes.

3. Access token managers to generate the requested tokens.

    1. Configure the access token manager instances. See Managing access token management instances.

    2. Map the attributes from the token exchange processor policies to the attributes from the access token manager instances. See Mapping token exchange attributes to access token manager attributes.

4. Enable token exchange in the OAuth clients that will send the token exchange requests to the authorization server. See Enabling token exchange in OAuth clients.

Defining token exchange processor policies

To exchange security tokens, the OAuth authorization server needs at least one token exchange processor policy.

Before you begin

Before you define a token exchange processor policy, create the necessary token processor instances. See Managing token processors.

About this task

In the **Token Exchange Processor Policy Management** window, configure and define a token exchange processor policy.

Steps

1. Go to **Applications** > **Token Exchange** > **Processor Polices** to open the**Token Exchange Processor Policy Management** window.

2. Click **Add Processor Policy**.

   Result:

   The **Token Exchange Processor Policy** window opens.

3. On the **Manage Processor Policy** tab, enter the policy **ID** and **Name**. Click **Next**.

   Select the **Actor Token Required** check box if you want to specify whether the policy requires an actor token as well as a subject token in the token exchange requests from the clients.

4. On the **Attribute Contract** tab, add attributes to the attribute contract as needed. Click **Next**.

5. On the **Token Processor Mapping** tab, map a token processor to each subject token type or each combination of subject token type and actor token type:

    1. Click the **Map New Token Processor** button.

       Result:

       The **Token Processor Mapping** window opens.

    2. On the **Token Types** tab, from the **Subject Token Processor** list, select the instance.

   3. In the **Subject Token Type** field, enter the identifier.

   4. If an actor token processor is required, from the **Actor Token Processor**list, select the instance.

   5. In the **Actor Token Type** field, enter the identifier. Click **Next**.

   6. On the **Attribute Sources & User Lookup** tab, add additional attribute sources for contract fulfillment as needed. Click **Next**.

   7. On the **Contract Fulfillment** tab, select the **Source** and **Value** for each attribute. Click **Next**.

   8. On the **Issuance Criteria** tab, specify conditions that attributes must satisfy for PingFederate to exchange the token. Click **Next**.

   9. On the **Summary** tab, review the token processor mapping. Click **Done**.

      ***Result:**PingFederate returns you to the **Token Exchange Processor Policy** window.*

6. On **Summary** tab, review the policy. Click **Done**.

   Result:

   The **Token Exchange Processor Policy Management** window opens.

7. If you want to make the new token exchange processor policy the default policy, click **Set as Default** on the corresponding row in the table.

8. Click **Save**.

Creating token exchange generator groups

A token exchange generator group maps requested token types to your token generator instances. You can create multiple token exchange generator groups. If you assign resource URIs to the groups, clients can use the URI in the resource parameter of its requests to specify a group.

Before you begin

Before you create a token exchange generator group, configure the token generator instances. See Managing token generators.

About this task

In the **Generator Groups** window, create a generator group for a token exchange instance.

Steps

1. Go to the **Applications** > **Token Exchange** > **Generator Groups** window.

2. Click the **Add Generator Group** button.

   The **Token Exchange Generator Group** window opens.

3. On the **Manage Generator Group** tab, enter the group **ID**, **Name**, and enter absolute **Resource URIs**.

4. On the **Requested Token Type Mapping** tab, from the **Token Generator** list, select an instance and enter the **Token Type**. Click **Add**.

5. Repeat step 4 for each type of token that you want the token exchange generator group to handle.

> **Important**
>
> If this is the default token exchange generator group and clients will use the `requested_token_type` parameter to request specific types of tokens, then map all token types that clients can request.

6. On the **Summary** tab, review the token exchange generator group. Click **Done**.

   The **Generator Groups** window opens.

7. If you want to make the new token exchange generator group the default group, click **Set as Default** in the **Action** column.

8. Click **Save**.

Mapping token exchange attributes to token generator attributes

When configuring the OAuth authorization server to exchange security tokens, if it uses token generator instances to create the requested tokens, then map the attributes in the attribute contract produced by the token exchange processor policy to the attributes created by the token generator instances.

Before you begin

Before you perform the following procedure:

- Define the token exchange processor policies. See Defining token exchange processor policies.
- Configure the token generator instances. See Managing token generators.

About this task

In the **Token Generator Mappings** window, map the attributes from a token exchange processor policy to the attributes from a token generator instance.

Steps

1. Go to **Applications** > **Token Exchange** > **Token Generator Mappings** to open the **Token Generator Mappings** window.

2. From the **Source Instance** list, select a token exchange processor policy.

3. From the **Target Instance** list, select a token generator from a token exchange generator group. Click **Add Mapping** button.

   Result:

   The **Mapping Configuration** window opens.

4. On the **Attribute Sources & User Lookup** tab, add token generators additional attribute sources for contract fulfillment as needed. Click **Next**.

5. On the **Token Contract Fulfillment** tab, select a **Source** and **Value** for each attribute. Click **Next**.

6. On the **Issuance Criteria** tab, add and specify conditions that attributes must satisfy for PingFederate to exchange the token as needed. Click **Next**.

7. On the **Summary** tab, review the mapping configuration. Click **Done**.

   Result:

The **Token Generator Mappings** window opens.

8. Click **Save**.

Mapping token exchange attributes to access token manager attributes

When configuring the OAuth authorization server to exchange security tokens, if it uses an access token manager instances to generate requested tokens, then map the attributes in the attribute contract produced by the token exchange processor policy to the attributes in the tokens created by the access token manager instances.

Before you begin

Before you perform the following procedure:

- Define the token exchange processor policies. See Defining token exchange processor policies.
- Configure the access token managers instances. See Managing access token management instances.

About this task

In the **Access Token Mapping** window, map the attributes from a token exchange processor policy to the attributes from an access token manager instance.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Mapping**.
2. In the **Context** section, from the **Context** list, select a token exchange processor policy.
3. From the **Access Token Manager** list, select an access token manager. Click **Add Mapping**.

   Result:

   The**Access Token Mapping** configuration window wizard opens.
4. On the **Attribute Sources & User Lookup**tab, add access token manager attribute sources for contract fulfillment as needed. Click **Next**.
5. On the **Contract Fulfillment** tab, select a **Source** and **Value** for each attribute. Click **Next**.
6. On the **Issuance Criteria** tab, add and specify conditions that attributes must satisfy for PingFederate to exchange the token as needed. Click **Next**.
7. On the **Summary** tab, review the access token mapping. Click **Done**.

   Result:

   The **Access Token Mapping** window opens.
8. Click **Save**.

Enabling token exchange in OAuth clients

After configuring the OAuth authorization server to exchange tokens, enable token exchange on each OAuth client that will send the authorization server token exchange requests.

Before you begin

Before you perform the following procedure, define the token exchange processor policy that the OAuth server uses when it receives a token exchange request from the client. See Defining token exchange processor policies.

About this task

In the **Client** window, enable OAuth token exchange in an OAuth client.

Steps

1. Go to **Applications** > **OAuth** > **Clients** to open the **Clients** window. Click the link of the client in the **Client ID** column.

   Result:

   The **Client** window opens.

2. In the **Allowed Grant Types** section, select the **Token Exchange** check box.

3. In the **Token Exchange** section, from the **Processor Policy** list, select the token exchange processor policy.

4. Click **Save**.

   Result:

   This will take you back to the **Clients** window.

5. Click **Save**.

OAuth rich authorization requests

PingFederate supports OAuth rich authorization requests.

The rich authorization request parameter, `authorization_details`, is used by some open banking and other deployments to carry fine-grained authorization data in OAuth messages.

Authorization details can be used in the same places where scope is used to specify authorization requirements. The following flows support authorization details:

- Authorization code
- Implicit
- Client Credentials
- Device Authorization
- CIBA
- Token Exchange (only available for mapping)

The `authorization_details` parameter is a JSON array of JSON objects, where `type` is the only required field for each object.

In the following example of an authorization detail, the `type` is `payment_initiation`:

```
[
  {
    "type": "payment_initiation",
    "locations": [
      "https://example.com/payments"
    ],
    "instructedAmount": {
      "currency": "EUR",
      "amount": "123.50"
    },
    "creditorName": "Merchant A",
    "creditorAccount": {
      "iban": "DE02100100109307118603"
    },
    "remittanceInformationUnstructured": "Ref Number Merchant"
  }
]
```

For more information about authorization details, see the OAuth 2.0 Rich Authorization Requests specification.

Configuring support for OAuth rich authorization requests

You can configure PingFederate to support rich authorization requests from OAuth clients.

Before you begin

Review the SDK documentation for the `com.pingidentity.sdk.authorizationdetails` package. You can find the SDK documentation in the `<pf_install>/pingfederate/sdk/doc` directory or in the compiled PingFederate Server SDK documentation.

Steps

- Use the PingFederate SDK to develop an authorization detail processor plugin that can process the authorization detail type you want to support.

> **Note**
>
> PingFederate does not include any authorization detail processors because handling the authorization details depends on your requirements and use cases.

Configuring authorization detail processors

Authorization detail processors are custom plugins you can develop and configure to support rich authorization requests.

Before you begin

You must have an authorization detail processor plugin that can process the authorization detail type you want to support.

About this task

To configure an authorization detail processor instance:

Steps

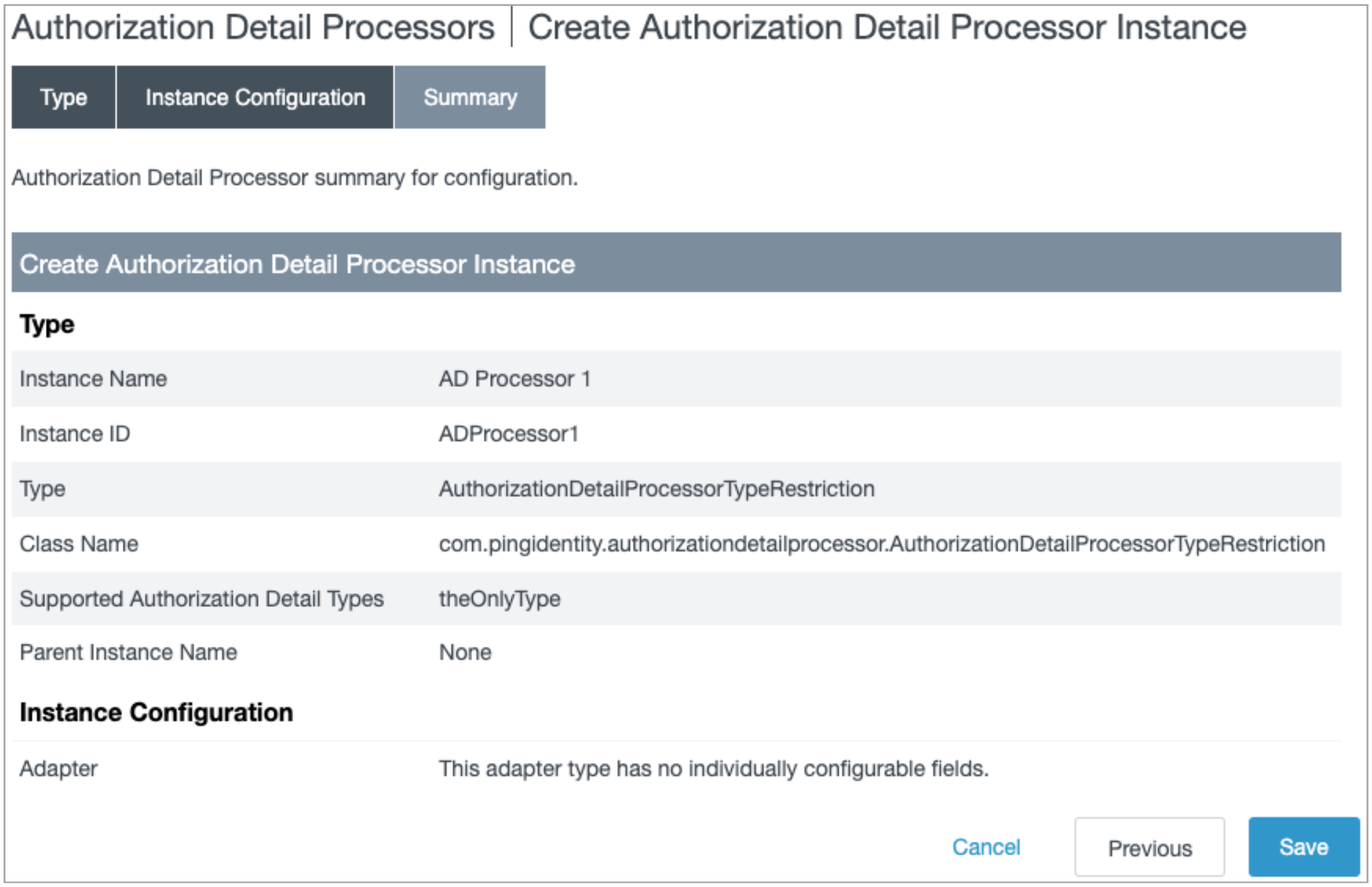
1. In PingFederate, go to **System > OAuth Settings > Authorization Detail Processors**.
2. In the **Authorization Detail Processors** window, click **Create New Instance**.

   Result:

   The **Create Authorization Detail Processor Instance** window opens.
3. On the **Type** tab:
   1. Enter an **Instance Name** and unique **Instance ID**.
   2. Select an **Authorization Detail Processor Type**.
   3. **Optional:** If other instances exist, select one of them as a **Parent Instance** on which to base this new instance.
4. On the **Instance Configuration** tab, configure the authorization detail processor instance.

   The tab's settings are determined by the plugin for the **Authorization Detail Processor Type** that you selected on the **Type** tab.
5. On the **Summary** tab, review the settings. Click **Save**.

> **Note**
>
> The **Summary** tab shows the **Class Name** and **Supported Authorization Details Types**, which are determined by your custom plugin.



Authorization Detail Processors | Create Authorization Detail Processor Instance
Type
Instance Configuration
Summary

Authorization Detail Processor summary for configuration.

Create Authorization Detail Processor Instance

Type

Instance Name	AD Processor 1
Instance ID	ADProcessor1
Type	AuthorizationDetailProcessorTypeRestriction
Class Name	com.pingidentity.authorizationdetailprocessor.AuthorizationDetailProcessorTypeRestriction
Supported Authorization Detail Types	theOnlyType
Parent Instance Name	None

Instance Configuration

Adapter	This adapter type has no individually configurable fields.

Configuring authorization detail types

To support rich authentication requests from OAuth clients, you must configure an authorization detail type for each type that you want PingFederate to support.

Before you begin

PingFederate must have an instance of an authorization detail processor that can process the authorization detail type that you're configuring.

> **Note**
>
> If you need information about the authorization detail processor instance when configuring an authorization detail type, go to the **System > OAuth Settings > Authorization Detail Processors** window, open the instance, and go to the **Create Authorization Detail Processor Instance** window and select the **Summary** tab.

About this task

To support rich authentication requests from OAuth clients, you must configure an authorization detail type for each `type` that you want PingFederate to support:

Steps

1. In PingFederate, go to **System > OAuth Settings > Authorization Detail Types**.
2. In the **Authorization Detail Types** window, click **Add Authorization Detail Type**.

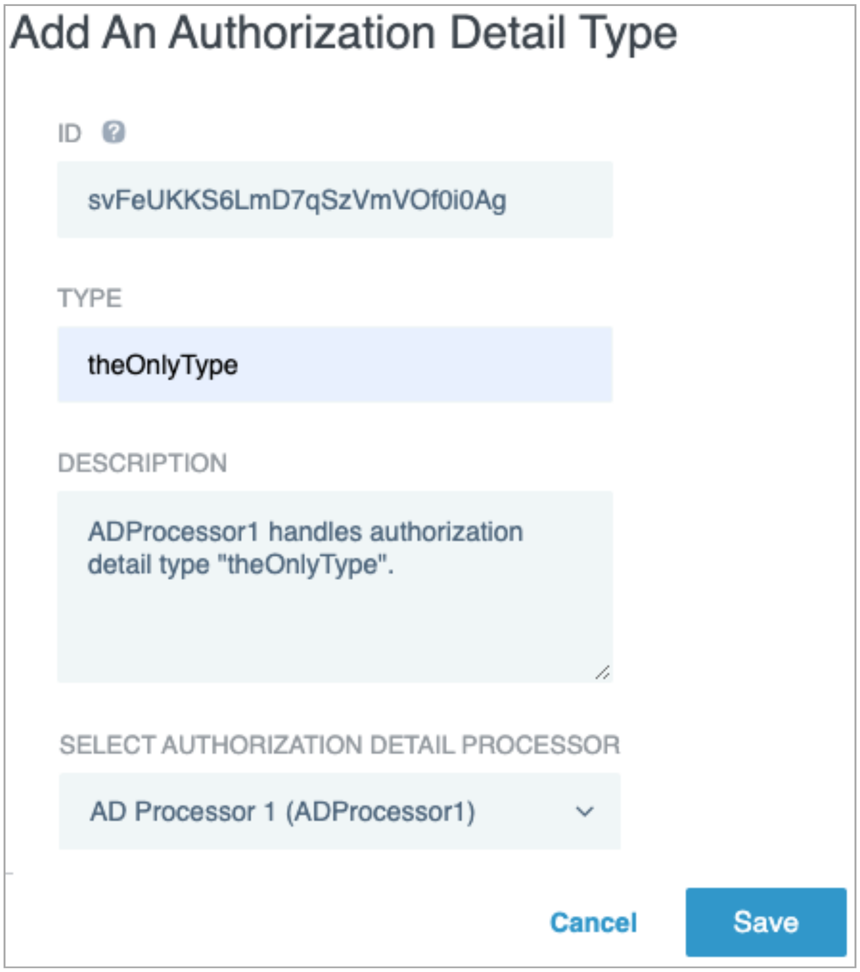
3. In the **Add an Authorization Detail Type** window:

    1. In the **Type** field, enter the name of a supported authorization detail type.

    2. Enter a **Description** for the authorization detail type.

    3. Select the **Authorization Detail Processor instance** to handle the authorization detail type.

    4. Click **Save**.



Add An Authorization Detail Type

ID

svFeUKKS6LmD7qSzVmVOf0i0Ag

TYPE

theOnlyType

DESCRIPTION

ADProcessor1 handles authorization detail type "theOnlyType".

SELECT AUTHORIZATION DETAIL PROCESSOR

AD Processor 1 (ADProcessor1)

Next steps

To complete your configuration:

1. Configure the external content user interface, adding an **External Consent Authorization Details Attribute**.

2. Configure the OAuth client, selecting the check boxes for **Allow Authorization Details** and the authorization detail type that you configured in the **Add an Authorization Detail Type** window in step 3 of this topic.

Security management

The **Security** menu provides access to security and infrastructure-related settings. Depending on the setup of PingFederate, menu items vary.

Security management consists of the following sections:

- Certificate and key management
- System integration
- Account lockout protection
- Password spraying prevention
- Implementing a MasterKeyEncryptor using AWS KMS

Certificate and key management

The PingFederate administrative console provides a suite of configuration wizards for administrators to manage keys and certificates.

Tasks include:

- Managing trusted certificate authorities (CAs)
- Managing server certificates for the administrative port and runtime ports
- Managing client certificates for mutual TLS authentication
- Managing signing and decryption keys and certificates
- Managing OAuth and OpenID Connect keys
- Managing certificates from partners
- Configuring certificate revocation settings
- Managing partner metadata URLs
- Rotating system keys

> **Note**
>
> For certificates that you own, you have two export options: certificate only or certificate and private key.
>
> - Certificate only - PingFederate exports in PEM format with the file extension `.pem`.
> - Certificate and private key - PingFederate exports in PEM or PKCS12 format with the file extension `.pem` or `.p12` respectively.
>
> For features that use a certificate that you own, you can either create a new certificate or import an existing PEM or PKCS12 certificate file.
> For partner certificates, you can only export the certificate. PingFederate exports the partner certificate in PEM format. You can also import a partner certificate in PEM format.
> If you are running in BCFIPS mode, you can only import and export in PEM format.

You can configure PingFederate to use a hardware security module (HSM) for cryptographic material storage and operations. When configured, private keys and their corresponding certificate are stored on the HSM. Related signing and decryption operations are processed there for enhanced security. By default, even in HSM mode, dynamic OAuth and OpenID Connect signing and decryption keys are generated and stored in the memory of PingFederate cluster nodes. To ensure continuity after a full cluster restart, the decryption keys are also persisted to disk, and encrypted there with PingFederate's active configuration encryption key. To ensure OAuth and OpenID Connect keys are instead stored on the HSM, you must enable static keys.

> ⓘ **Note**
>
> Management of keys and certificates is restricted to administrative users with the **Crypto Admin** administrative role (see Administrative accounts).

See subsequent topics for configuration steps.

Manage trusted certificate authorities

On the **Trusted CAs** window, you can import, export, review, and remove certificate authorities (CAs).

You can import your federation partner's CA certificate or self-signed certificates into PingFederate's global trust list on **Security > Certificate & Key Management > Trusted CAs**. If the CA is not one of the major authorities, you might also need to import the certificate from the CA that signed the partner certificate.

> ⓘ **Note**
>
> If a required CA certificate is already available from the Java runtime, you do not need to import the same certificate into the PingFederate store.

Importing trusted certificate authorities

Import your federation partner's certificate authority (CA) certificate or self-signed certificates into PingFederate's global trust list.

Steps

1. On the **Trusted CAs** window, click **Import**.
2. On the **Import Certificate** window, choose the applicable certificate file.

   > ⓘ **Note**
   >
   > If PingFederate is integrated with a hardware security module (HSM) from Thales in hybrid mode, select the storage facility of the certificate from the **Cryptographic Provider** list.
   >
   > - Select **HSM** to store the certificate in the HSM.
   > - Select **Local Trust Store** to store the certificate in the local trust store managed by PingFederate.

3. On the**Summary** window, review your configuration, amend as needed, and click **Save**.

Exporting trusted certificate authorities

Export your federation partner's certificate authority (CA) certificate or self-signed certificates as desired.

Steps

1. On the **Trusted CAs** window, select **Action** > **Export** for the certificate.
2. On the **Export Certificate** window, click **Next**.
3. On the **Export & Summary** window, click **Export** to save the certificate file and then click **Done**.

Reviewing trusted certificate authorities

Review certificates to ensure you've selected the correct ones.

Steps

1. On the **Trusted CAs** window, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.
3. When finished, close the pop-up window.

Removing trusted certificate authorities

Remove certificates from the **Trusted CAs** window when necessary.

Steps

1. On the **Trusted CAs** window, select **Action** > **Delete** for the certificate.

   > **Note**
   >
   > To cancel the removal request, select **Action** > **Undelete** for the certificate.

2. Click **Save** to confirm your action.

Manage SSL server certificates

On the **Security** > **Certificate & Key Management** > **SSL Server Certificates** window, you can establish and maintain the certificates presented for access to the PingFederate administrative console (or the administrative API) and for incoming HTTPS connections at runtime.

The first system-generated certificate is the default certificate for both the administrative console and the runtime server. As multiple certificates are created, you can activate or deactivate them for the administrative console, the runtime server, or both. Additionally, you can select any of them as the new default certificate for the administrative console, the runtime server, or both at a later time.

When creating a certificate, you can add additional domain names through the use of the **Subject Alternative Names** field. Furthermore, if a user agent includes the host name that it intends to reach as part of the TLS handshake, PingFederate selects the applicable certificate based on the provided Server Name Indication (SNI) information. The selection looks at the common name and subject alternative names of each activated certificate. If PingFederate finds no match, it serves the default certificate. If PingFederate finds multiple matches, it serves the certificate with the better match.

Note

If PingFederate finds multiple certificates of the same matching quality, it returns one of them in the TLS handshake. This response should not impact the user agent because either the common name or one of the subject alternative names matches the host name of the request. If PingFederate should always serve a particular certificate for any given host name, ensure that the common name and any configured subject alternative names do not overlap among multiple certificates.

SSL Server Certificates configuration

Certificate	Common name	Subject alternative names	Activation status
#1	`www.example.com`	(None)	Administrative console and runtime server
#2	`www.example.org`	`*.example.org` and `test.example.local`	Administrative console and runtime server
#3	`www.example.info`	**`.example.info`** and `.example.com`	Administrative console and runtime server
#4	`admin.example.local`	(None)	Administrative console (Default) and runtime server
#5	`runtime.example.local`	(None)	Administrative console and runtime server (Default)

Runtime behavior

Request type	Host name from SNI	Certificate served
Administrative or runtime	www.example.com	The host name from the SNI is an exact match to the common name of certificate #1 and a partial match to the second subject alternative name (`*.example.org`) of certificate #3. An exact match is a better match, so PingFederate serves certificate #1.
Administrative or runtime	www.example.org	The host name from the SNI is an exact match to the common name of certificate #2. PingFederate serves certificate #2.
Administrative or runtime	sso.example.org	The host name from the SNI is a partial match to the first subject alternative name (`*.example.org`) of certificate #2. There is no other exact or partial match. PingFederate serves certificate #2.

Request type	Host name from SNI	Certificate served
Administrative or runtime	sso.example.info	The host name from the SNI is a partial match to the first subject alternative name (`*.example.info`) of certificate #3. There is no other exact or partial match. PingFederate serves certificate #3.
Administrative or runtime	sso.example.com	The host name from the SNI is a partial match to the second subject alternative names (`*.example.com`) of certificate #3. There is no other exact or partial match. PingFederate serves certificate #3.
Administrative	www.example.local	The host name from the SNI does not match any configured certificate. PingFederate serves certificate #4, the default certificate for the administrative console.
Runtime	localhost	The host name from the SNI does not match any configured certificate. PingFederate serves certificate #5, the default certificate for the runtime server.

Creating a new certificate

On the **SSL Server Certificates** window, you can generate customized certificates.

Steps

1. On the **SSL Server Certificates** window, click **Create new**.
2. On the **Create Certificate** window, enter the required information.

   For information about each field, refer to the following table.

Field	Description
Common Name	The common name (CN) identifying the certificate.
Subject Alternative Names	The additional DNS names or IP addresses that can be associated with the certificate.
Organization	The organization (O) or company name creating the certificate.
Organizational Unit	The specific unit within the organization (OU).
City	The city or other primary location (L) where the company operates.
State	The state (ST) or other political unit encompassing the location.
Country	The country © where the company is based.
Validity (days)	The time during which the certificate is valid.

Field	Description
Cryptographic Provider	The storage facility of the certificate. Applicable and visible only when PingFederate is integrated with an HSM in hybrid mode. ◦ Select **HSM** to store the certificate in the HSM. ◦ Select **Local Trust Store** to store the certificate in the local trust store managed by PingFederate.
Key Algorithm	A cryptographic formula used to generate a key. PingFederate uses either of two algorithms, RSA or EC.
Key Size (bits)	The number of bits used in the key. (RSA-1024, 2048 and 4096; and EC-256, 384 and 521.)
Signature Algorithm	The signing algorithm of the certificate. (RSA-SHA256, SHA384, and SHA512; and ECDSA-SHA256, SHA384, and SHA512.)

3. When finished, click **Next**.

4. On the **Summary** window, review your configuration, amend as needed, and click **Save**.

Importing a certificate and its private key

You can import certificates and their private keys in the **SSL Server Certificates** window.

About this task

This task describes how to import certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
  - In non-BCFIPS mode, we support PKCS12 and PEM formatted certificates and private keys, and automatically detect the format between PKCS12 and PEM.
  - In BCFIPS mode, we only support PEM formatted certificate and private keys. Only PBES2 and AES or Triple DES encryption is accepted and 128-bit salt is required. In practice, this may mean that only PEM files generated by PingFederate can be imported.
  - For PEM, the private key must precede the certificates.
- Password requirement:
  - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **SSL Server Certificates** window, click **Import**.

2. On the **Import Certificate** window, choose the applicable certificate file and enter its password.

> **Note**
>
> If PingFederate is integrated with a hardware security module (HSM) from Thales, you cannot use an elliptic curve (EC) certificate as an SSL server certificate. You must select a certificate that uses the RSA key algorithm.

3. If PingFederate is integrated with an HSM in hybrid mode, select the storage facility of the certificate from the **Cryptographic Provider** list.
    1. Select **HSM** to store the certificate in the HSM.
    2. Select **Local Trust Store** to store the certificate in the local trust store managed by PingFederate.
4. On the **Summary** window, review your configuration, amend as needed, and click **Save**.

Creating a certificate-authority signing request (CSR)

On the **SSL Server Certificates** window, you can generate a CSR file for a certificate.

Steps

1. On the **SSL Server Certificates** window, select **Action > Certificate Signing** for the certificate.

> **Note**
>
> This selection is inactive if you have not yet saved a newly created or imported certificate. Click **Save** and then return to this window to initiate the process.
> The selection is also inactive if a previously signed certificate has been revoked. Because the revocation may indicate that the private key has been compromised, the best practice is to import or create a replacement certificate for certificate signing.

2. On the **Certificate Signing** window, select the **Generate CSR** option.
3. On the **Generate CSR** window, click **Export** to save the CSR file, and then click **Done**.

> **Note**
>
> Once saved, you can submit this CSR file to a certificate authority (CA) for a CA-signed certificate.

Importing a certificate-authority response (CSR response)

On the **SSL Server Certificates** window, you can import CSR response files for certificates.

Steps

1. On the **SSL Server Certificates** window, select **Action > Certificate Signing** for the certificate.
2. On the **Certificate Signing** tab, select the **Import CSR Response** option. Click **Next**.
3. On the **Import CSR Response** tab, click **Choose File**, and select the applicable CSR response file. Click **Next**.
4. On the **Summary** tab, review your configuration, and click **Save**.

Exporting a certificate

On the **SSL Server Certificates** window, you can export a certificate with or without its private key.

About this task

About this task

This task describes how to export certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
  - In non-BCFIPS mode, when the **Certificate and Private Key** option is selected, a **Format** field displays allowing you to choose between exporting a PKCS12 or a PEM formatted certificate and private key.
  - In BCFIPS mode, you can only export PEM-formatted certificates and private keys.

    If you need to convert from PEM to PKCS12 format, use the following command:

    ```
    openssl pkcs12 -export -inkey keypair.pem -in keypair.pem -out keypair.p12
    ```
- Password requirement:
  - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **SSL Server Certificates** window, select **Action > Export** for the certificate.
2. On the **Export Certificate** window, select the export type.
   - Select **Certificate Only** to export the selected certificate without its private key. This is the default choice.
   - Select **Certificate and Private Key** to export the selected certificate with its private key. If you are *not* running in BCFIPS mode, the **Format** section appears, and you must select either **PKCS12** or **PEM**.

     You must also enter and confirm an **Encryption Password**, since this export contains the private key of the certificate.

   If the selected certificate is stored in a hardware security module (HSM), the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** window, click **Export** to save the certificate file, and then click **Done**.

Reviewing a certificate

On the **SSL Server Certificates** window, you can review a particular certificate.

Steps

1. On the **SSL Server Certificates** window, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.
3. When finished, close the pop-up window.

Activating or deactivating a certificate

On the **SSL Server Certificates**, you can configure whether to activate or deactivate a certificate.

Steps

1. On the **SSL Server Certificates** window, select the relevant option under **Action** for the certificate.

   Any certificate can be activated for the administrative console, the runtime server, or both. When multiple certificates are activated for the administrative console (or the runtime server), you can deactivate any of them as long as one certificate remains active. Additionally, you may select any of them as the default certificate.

2. Click **Save** to keep your configuration.

Removing a certificate

On the **SSL Server Certificates** window, you can delete unwanted certificates.

Steps

1. On the **SSL Server Certificates** window, select **Action** > **Delete** for the certificate.

   > **Note**
   >
   > If the selected certificate is activated for the administrative port, the runtime port, or both, the **Delete** option does not apply.
   > To cancel the removal request, select **Action** > **Undelete** for the certificate.

2. Click **Save** to confirm your action.

Manage SSL client keys and certificates

On **Security** > **Certificate & Key Management** > **SSL Client Keys & Certificates**, you can create and manage your authentication private keys and the certificates your server presents as clients in an outbound SSL/TLS transaction.

The **SSL Client Keys & Certificates** window enables you to manage certificates and CSRs in multiple ways. The window's functionality allows you to create, import, export, review, and delete certificates, as well as create CSRs and import CSR responses.

Creating new certificates

Use the functionality found in the **SSL Client Keys & Certificates** window to create new, customized certificates.

Steps

1. On the **SSL Client Keys & Certificates** window, click **Create new**.

2. On the **Create Certificate** tab, enter the required information.

   For information about each field, refer to the following table.

Field	Description
Common Name	The common name (CN) identifying the certificate.
Subject Alternative Names	The additional DNS names or IP addresses possibly associated with the certificate.
Organization	The organization (O) or company name creating the certificate.
Organizational Unit	The specific unit within the organization (OU).
City	The city or other primary location (L) where the company operates.
State	The state (ST) or other political unit encompassing the location.
Country	The country © where the company is based.
Validity (days)	The time during which the certificate is valid.
Key Algorithm	A cryptographic formula used to generate a key. PingFederate uses either of two algorithms, RSA or EC.
Key Size (bits)	The number of bits used in the key. (RSA-1024, 2048 and 4096; and EC-256, 384 and 521.)
Signature Algorithm	The signing algorithm of the certificate. (RSA and ECDSA-SHA256, SHA384, and SHA512.)

3. When finished, click **Next**.

4. On the **Summary** tab, review your configuration, amend as needed, and click **Done**.

Importing certificates and their private keys

You can import certificates and their private keys in the **SSL Client Keys & Certificates** window.

About this task

This task describes how to import certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
  - In non-BCFIPS mode, we support PKCS12 and PEM formatted certificates and private keys, and automatically detect the format between PKCS12 and PEM.
  - In BCFIPS mode, we only support PEM formatted certificate and private keys. Only PBES2 and AES or Triple DES encryption is accepted and 128-bit salt is required. In practice, this may mean that only PEM files generated by PingFederate can be imported.
  - For PEM, the private key must precede the certificates.
- Password requirement:
  - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **SSL Client Keys & Certificates** window, click **Import**.
2. On the **Import Certificate** tab, choose the applicable certificate file and enter its password.

> **Note**
>
> If PingFederate is integrated with an HSM in hybrid mode, select the storage facility of the certificate from the **Cryptographic Provider** list.
>
> - Select **HSM** to store the certificate in the HSM.
> - Select **Local Trust Store** to store the certificate in the local trust store managed by PingFederate.

3. On the **Summary** window, review your configuration, amend as needed, and click **Done**.

Creating a certificate signing request (CSR)

Use the **Certificate Signing** functionality to generate and save a CSR file to submit it to a certificate authority (CA) for a signed certificate.

Steps

1. On the **SSL Client Keys & Certificates** window, select **Certificate Signing** for the certificate.

> **Note**
>
> This selection is inactive if you have not yet saved a newly created or imported certificate. Click **Save** and then return to this window to initiate the process.
> The selection is also inactive if a previously signed certificate is revoked. Because the revocation could indicate that the private key is compromised, the best practice is to import or create a replacement certificate for certificate signing.

2. On the **Certificate Signing** tab, select the **Generate CSR** option.
3. On the **Generate CSR** tab, click **Export** to save the CSR file, and then click **Done**.

> **Note**
>
> Once saved, you can submit this CSR file to a certificate authority for a CA-signed certificate.

Importing a certificate-authority response (CSR response)

Use the **Certificate Signing** functionality to import your own CSR response file into PingFederate.

Steps

1. On the**SSL Client Keys & Certificates** window, select **Certificate Signing** for the certificate.
2. On the **Certificate Signing** tab, select the **Import CSR Response** option.
3. On the **Import CSR Response** tab, choose the applicable CSR response file.
4. On the **Summary** tab, review your configuration, and click **Save**.

Exporting certificates

On the **SSL Client Keys & Certificates** window, you can export a certificate with or without its private key.

About this task

About this task

This task describes how to export certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
  - In non-BCFIPS mode, when the **Certificate and Private Key** option is selected, a **Format** field displays allowing you to choose between exporting a PKCS12 or a PEM formatted certificate and private key.
  - In BCFIPS mode, you can only export PEM-formatted certificates and private keys.

    If you need to convert from PEM to PKCS12 format, use the following command:

    ```
    openssl pkcs12 -export -inkey keypair.pem -in keypair.pem -out keypair.p12
    ```
- Password requirement:
  - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **SSL Client Keys & Certificates** window, select **Export** for the certificate.
2. On the **Export Certificate** tab, select the export type.
   - Select **Certificate Only** to export the selected certificate without its private key. This is the default choice.
   - Select **Certificate and Private Key** to export the selected certificate with its private key. If you are *not* running in BCFIPS mode, the **Format** section appears, and you must select either **PKCS12** or **PEM**.

     You must also enter and confirm an **Encryption Password**, since this export contains the private key of the certificate.

   If the selected certificate is stored in a hardware security module (HSM), the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** window, click **Export** to save the certificate file, and then click **Done**.

Reviewing certificates

Take a closer look at individual certificates to ensure their properties match your needs.

Steps

1. On the **SSL Client Keys & Certificates** window, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.
3. When finished, close the pop-up window.

Removing certificates

Delete certificates you no longer need.

Steps

1. On the **SSL Client Keys & Certificates** window, select **Delete** for the certificate.

   > **Note**
   >
   > To cancel the removal request, select **Undelete** for the certificate.

2. Click **Save** to confirm your action.

Manage digital signing certificates and decryption keys

On **Security** > **Certificate & Key Management** > **Signing & Decryption Keys & Certificates**, you can create and maintain certificates and their respective key pairs for the purpose of signing outgoing requests, responses, assertions, and access tokens, and for the purpose of decryption.

Use separate certificates for signing and decryption.

After creating your certificates, if they remain as self-signed certificates, you can enable automatic certificate rotation.

Automatic certificate rotation

The optional automatic certificate rotation feature of PingFederate greatly reduces the cost of managing self-signed certificates.

PingFederate supports automatic certificate rotation for self-signed certificates created for signing SAML requests, responses, and assertions, or XML decryption for browser SSO and WS-Trust STS transactions on a per-certificate basis.

> **Note**
>
> Certificate rotation is only available to self-signed certificates. Also, you can't enable rotation on certificates that are used as a secondary signing certificate in a connection, or are used as the primary certificate in a connection configured with a secondary signing certificate.

Certificate rotation happens over two stages, identified by the **Creation Buffer** and **Activation Buffer** settings.

- The **Creation Buffer** is the number of days ahead of expiry that PingFederate creates a new key pair and a new certificate.
- The **Activation Buffer** is the number of days ahead of expiry that PingFederate activates the certificate.

When you enable certificate rotation on a certificate, you can customize the values of the **Creation Buffer** and **Activation Buffer** settings. Alternatively, you can keep their default values, which are 25% and 10% of the original lifetime of the current certificate, respectively. The following examples illustrate the default values for both buffers based on a 100-day certificate and a 365-day certificate.

Current certificate	The default value for theCreation Bufferfield	The default value for theActivation Bufferfield	The rotation window
Self-signed certificate #1, valid for 100 days from January 1, 2017 to April 9, 2017	25 days ahead of expiry, which is March 16	10 days ahead of expiry, which is March 31	15 days from March 16 through March 30
Self-signed certificate #2, valid for 365 days from January 1, 2017 to December 31, 2017	91 days ahead of expiry, which is October 2	36 days ahead of expiry, which is November 26	55 days from October 2 through November 25

If the PingFederate server is shut down when the **Creation Buffer** threshold is reached for a given certificate, a new key pair and a new certificate are created if PingFederate is restarted during the rotation window.

In a clustered PingFederate environment, when the new signing certificate is ready, the administrative console displays a message to remind the administrators to replicate the new certificate to the engine nodes in **System > Server > Cluster Management**.

Although optional, you can turn on notifications for certificate events in **System > Monitoring & Notifications > Runtime Notifications**. When configured, PingFederate notifies the configured recipient when a new certificate is available and when it is activated. Depending on the role of the certificate, you can update your partner accordingly.

Connection and federation metadata

Certificate rotation uses a number of inherent capabilities which enable it to deploy new certificates to replace current certificates in enabled connections.

Certification rotation is a per-certificate configuration. When certificate rotation is enabled for a certificate and a new certificate using new key pairs becomes available, PingFederate deploys the new certificate to all enabled connections that use the original certificate. The actions taken by PingFederate vary depending on the role of the certificate.

Notifications

Although optional, you can turn on notifications for certificate events in **System > Monitoring & Notifications > Runtime Notifications**. When configured, PingFederate notifies the configured recipient when a new certificate is available and when it is activated. Depending on the role of the certificate, you can update your partner accordingly.

Signing certificate

When the **Creation Buffer** threshold is reached, a new certificate is created. For all web browser single sign-on (SSO) (SAML and WS-Federtion) connections using the same signing certificate, PingFederate starts including the new certificate (along with the current certificate) in their metadata. PingFederate keeps using the current certificate for signing until the remaining lifetime of the current certificate reaches the **Activation Buffer** threshold, at which point PingFederate starts signing with the new certificate and removes the previous certificate from the metadata.

> **Important**
>
> To prevent SSO outages, partners must update their connections to use the new certificate to verify digital signatures before the **Activation Buffer** threshold is reached.

XML decryption

When a new certificate is made available, PingFederate performs the following tasks for all SAML 2.0 connections using the same decryption key:

- Pushes the current decryption key from primary to secondary
- Places the new certificate as the primary decryption key
- Updates the decryption key with the new certificate in the metadata
- Starts using the new decryption key to decrypt inbound messages. If the primary decryption key fails, PingFederate fails over to the secondary decryption key

When the remaining lifetime of the current certificate reaches the **Activation Buffer** threshold, the secondary decryption key is removed from the SAML 2.0 connections.

When PingFederate is configured to generate notifications for certificate events, PingFederate also notifies the configured recipient when the existing RSA decryption key is about to expire.

> **Important**
>
> For XML decryption keys, PingFederate only supports the RSA key algorithm. When **EC** (elliptic curve) is selected as the **Key Algorithm** value on the **Certificate Rotation** tab, PingFederate does not update the SAML 2.0 connections and their metadata.

> **Important**
>
> To prevent SSO outages, partners must update their connections to use the new certificate to encrypt messages before the **Activation Buffer** threshold is reached.

Federation metadata for Browser SSO connections

PingFederate updates the metadata for the applicable web browser SSO connections as soon as a new certificate is available.

To ensure that your partners are aware of the new certificate, you can provide their respective federation metadata by URLs or exports.

Metadata by URL

PingFederate runtime engine provides an endpoint (`/pf/federation_metadata.ping`) to return metadata for web browser SSO connections. A service provider (SP) or an identity provider (IdP) is identified by its entity IDs using the `PartnerSpId` query parameter or the `PartnerIdpId` query parameter, respectively, as illustrated in the following examples.

Partner	Federation metadata URL to be given to the partner
An SP partner with an entity ID of SP1.	https://www.example.com:9031/pf/federation_metadata.ping? `PartnerSpId` =SP1
An IdP partner with an entity ID of IdP1.	https://www.example.com:9031/pf/federation_metadata.ping? `PartnerIdpId` =IdP1

ⓘ **Note**

The base URL for the PingFederate runtime engine is https://www.example.com:9031

Important

In a clustered environment, because the console node is responsible for creating and applying the new certificates to all applicable connections, you must replicate the new certificate to the engine nodes in **System > Server > Cluster Management** when the new certificate is available, so that the federation metadata for these connections is updated accordingly.

The administrative console reminds you to replicate configuration when it detects configuration changes.

Metadata by manual export

Alternatively, you can export a metadata file for a connection from the **Connections Management** window or **System > Protocol Metadata > Metadata Export**.

ⓘ **Note**

PingFederate does not deploy new certificates or update metadata for inactive connections.

WS-Trust STS connections

For connections with only the WS-Trust security token service (STS) profile, you must export the new pending certificate and pass it to your partners out-of-band before the **Activation Buffer** threshold is reached.

If a connection contains both the Browser SSO and the WS-Trust STS profiles, the new certificate is included in the federation metadata for the Web Browser SSO profile. Your partner can reuse the certificate from the metadata by URL or manual export and apply it to its STS configuration.

Managing certificate rotation settings

Use the **Signing & Decryption Keys & Certificates** window to customize certificate rotation settings for your certificates.

About this task

Manage certificate rotation settings for self-signed certificates on **Security > Certificate & Key Management > Signing & Decryption Keys & Certificates**.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select **Certificate Rotation** for the applicable certificate.

   ⓘ **Note**

   Certificate rotation is only available to self-signed certificates.

2. Select the check box to turn on certificate rotation for the selected certificate, then click **Next**.

   If you want to turn off certificate rotation for the selected certificate, clear the check box and then click **Save**.

3. **Optional:** On the **Certificate Rotation** tab, modify the default values.

Field	Description
Creation buffer	The number of days ahead of expiry that PingFederate creates a new key pair and a new certificate. The default value is 25% of the original lifetime of the current certificate.
Activation buffer	The number of days ahead of expiry that PingFederate activates the certificate. The default value is 10% of the original lifetime of the current certificate.
Validity	The time during which the certificate is valid. The default value matches that of the current certificate.
Key Algorithm	A cryptographic formula used to generate a key. PingFederate uses either of two algorithms, RSA or EC. The default value matches that of the current certificate. **Important** For XML decryption keys, PingFederate only supports the RSA key algorithm. When **EC** (elliptic curve) is selected as the **Key Algorithm** value on the **Certificate Rotation** tab, PingFederate does not update the SAML 2.0 connections and their metadata.
Key Size	The number of bits used in the key. (RSA-1024, 2048 and 4096; and EC-256, 384 and 521.) The default value matches that of the current certificate.
Signature Algorithm	The signing algorithm of the certificate. (RSA and ECDSA-SHA256, SHA384 and SHA512.) The default value matches that of the current certificate.

4. On the **Certificate Rotation Summary** tab, review the rotation settings. Adjust as needed, and then click **Save** to turn on automatic certificate rotation for this certificate.

Managed SP connection to PingOne for Enterprise and signing certificate

Use managed service provider (SP) connections to PingOne for Enterprise to automatically rotate signing certificates being used by it.

PingFederate automatically rotates the signing certificate used by the managed SP connection to PingOne for Enterprise.

> **Note**
>
> A managed SP connection to PingOne for Enterprise is a connection created either as part of the initial setup or the **System > External Systems > Connect to PingOne for Enterprise** configuration wizard in PingFederate 8.0 or later.

The certificate rotation settings are as follows.

Field	Values
Creation Buffer (days)	90

Field	Values
Activation Buffer (days)	30
Validity (days)	1095
Key Algorithm	RSA
Key Size	2048
Signature Algorithm	RSA SHA256

If the signing certificate should be manually rotated instead, disable automatic certificate rotation. See Managing certificate rotation settings.

> **Note**
>
> After making changes, the administrative console prompts for confirmation whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise in a banner message. See Managing PingOne for Enterprise settings.

Creating new certificates

Use the functionality found in the **Signing & Decryption Keys & Certificates** window to create new, customized certificates.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, click **Create new**.
2. On the **Create Certificate** tab, enter the required information.

   For information about each field, refer to the following table.

Field	Description
Common Name	The common name (CN) identifying the certificate.
Subject Alternative Names	The additional DNS names or IP addresses possibly associated with the certificate.
Organization	The organization (O) or company name creating the certificate.
Organizational Unit	The specific unit within the organization (OU).
City	The city or other primary location (L) where the company operates.
State	The state (ST) or other political unit encompassing the location.
Country	The country © where the company is based.
Validity (days)	The time during which the certificate is valid.

Field	Description
Key Algorithm	A cryptographic formula used to generate a key. PingFederate uses either of two algorithms, RSA or EC.
Key Size (bits)	The number of bits used in the key. (RSA-1024, 2048 and 4096; and EC-256, 384 and 521.)
Signature Algorithm	The signing algorithm of the certificate. (RSA and ECDSA-SHA256, SHA384, and SHA512.)

3. When finished, click **Next**.

4. On the **Summary** window, review your configuration, amend as needed, and click **Done**.

Importing certificates and their private keys

You can import certificates and their private keys in the **Signing & Decryption Keys & Certificates** window.

About this task

This task describes how to import certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
    - In non-BCFIPS mode, we support PKCS12 and PEM formatted certificates and private keys, and automatically detect the format between PKCS12 and PEM.
    - In BCFIPS mode, we only support PEM formatted certificate and private keys. Only PBES2 and AES or Triple DES encryption is accepted and 128-bit salt is required. In practice, this may mean that only PEM files generated by PingFederate can be imported.
    - For PEM, the private key must precede the certificates.
- Password requirement:
    - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, click **Import**.

2. On the **Import Certificate** tab, choose the applicable certificate file and enter its password.

> **Note**
>
> If PingFederate is integrated with an HSM in hybrid mode, select the storage facility of the certificate from the **Cryptographic Provider** list.
>
> - Select **HSM** to store the certificate in the HSM.
> - Select **Local Trust Store** to store the certificate in the local trust store managed by PingFederate.

3. On the **Summary** window, review your configuration, amend as needed, and click **Done**.

Creating a certificate signing request (CSR)

Use the **Certificate Signing** functionality to generate and save a CSR file to submit it to a certificate authority (CA) for a signed certificate.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select **Certificate Signing** for the certificate.

   > **Note**
   >
   > This selection is inactive if you have not yet saved a newly created or imported certificate. Click **Save** and then return to this window to initiate the process.
   > The selection is also inactive if a previously signed certificate is revoked. Because the revocation could indicate that the private key is compromised, the best practice is to import or create a replacement certificate for certificate signing.

2. On the **Certificate Signing** tab, select the **Generate CSR** option.

3. On the **Generate CSR** tab, click **Export** to save the CSR file, and then click **Done**.

   > **Note**
   >
   > Once saved, you can submit this CSR file to a certificate authority for a CA-signed certificate.

Importing a certificate-authority response (CSR response)

Use the **Certificate Signing** functionality to import your own CSR response file into PingFederate.

Steps

1. On the**Signing & Decryption Keys & Certificates** window, select **Certificate Signing** for the certificate.

2. On the **Certificate Signing** tab, select the **Import CSR Response** option.

3. On the **Import CSR Response** tab, choose the applicable CSR response file.

4. On the **Summary** tab, review your configuration, and click **Save**.

Exporting certificates

On the **Signing & Decryption Keys & Certificates** window, you can export a certificate with or without its private key.

About this task
About this task

This task describes how to export certificates and their private keys. Supported certificate and private key formats differ depending on whether you are running PingFederate with BCFIPS enabled or disabled.

- Certificate and private key format:
  - In non-BCFIPS mode, when the **Certificate and Private Key** option is selected, a **Format** field displays allowing you to choose between exporting a PKCS12 or a PEM formatted certificate and private key.
  - In BCFIPS mode, you can only export PEM-formatted certificates and private keys.

If you need to convert from PEM to PKCS12 format, use the following command:

```
openssl pkcs12 -export -inkey keypair.pem -in keypair.pem -out keypair.p12
```

- Password requirement:
  - In BCFIPS mode, the password must contain at least 14 characters.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select **Export** for the certificate.
2. On the **Export Certificate** tab, select the export type.
   - Select **Certificate Only** to export the selected certificate without its private key. This is the default choice.
   - Select **Certificate and Private Key** to export the selected certificate with its private key. If you are *not* running in BCFIPS mode, the **Format** section appears, and you must select either **PKCS12** or **PEM**.

     You must also enter and confirm an **Encryption Password**, since this export contains the private key of the certificate.

   If the selected certificate is stored in a hardware security module (HSM), the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** window, click **Export** to save the certificate file, and then click **Done**.

Reviewing certificates

Take a closer look at individual certificates to ensure their properties match your needs.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.
3. When finished, close the pop-up window.

Reviewing a certificate's usage

Take a look at a certificate's usage data to get a sense of how often it's used.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select **Check Usage** for the certificate.

   > **Note**
   >
   > If the certificate is not used by any configuration, the **Check Usage** option does not apply.
2. Review the information in the pop-up window.
3. When finished, close the pop-up window.

Removing certificates

Delete certificates you no longer need.

Steps

1. On the **Signing & Decryption Keys & Certificates** window, select **Delete** for the certificate.

   > **Note**
   >
   > To cancel the removal request, select **Undelete** for the certificate.

2. Click **Save** to confirm your action.

Keys for OAuth and OpenID Connect

You can use keys to manage a number of security roles in PingFederate.

On **Security > Certificate & Key Management > OAuth & OpenID Connect Keys**, you can specify whether PingFederate should use static or dynamically rotating keys for OAuth and OpenID Connect.

> **Note**
>
> When using dynamically rotating keys, the number of key sets in memory is set to three for both signing and encryption keys. This number is not configurable. The key sets include pending, active, and retired. At each rotation cycle, a new set of pending keys is generated. The original pending set becomes the active set, the active set becomes the retired set, and the old retired set goes away. All three sets are published for signing keys. For encryption keys, only the active key set is published. The rotation period and RSA key size are configurable in the file `<pf_install>/pingfederate/server/default/data/config-store/jwks-endpoint-configuration.xml` .

The keys are used in the following manner.

PingFederate role	Key usages
Authorization Server (AS)	Sign self-contained access tokens for relying parties (RPs).
OpenID Provider (OP)	Sign ID tokens for RPs.
Relying Party (RP)	Sign JSON web tokens (JWTs) for authentication, sign OpenID Connect request objects, decrypt ID tokens, or any combination.

JSON Web Keys endpoint

PingFederate supports the OpenID Connect (OIDC) JSON Web Key Set (JWKS) endpoint.

This endpoint provides a list of JSON Web Keys (JWKs) used to validate, sign, and encrypt JSON Web Token (JWT). The endpoint does not require client authentication and includes configured Elliptic Curve (EC) and RSA-based keys.

Each key listed in the endpoint includes a key ID (`kid`) and other claims. When PingFederate generates and signs a JWT, it includes the matching `kid` in the JWT header to identify the public key required to verify the signature.

After downloading the JWKS, a client can locate the appropriate public key by matching the `kid` and then validating the JWT signature. To encrypt a JWT, a client uses a JWK whose usage is configured for encryption and includes the `kid` in the generated JWT header.

The following table outlines the property names and descriptions of the JWKS endpoint.

Property name	Description
`kty`	The family of cryptographic algorithms used with the key.
`kid`	The unique identifier for the key.
`use`	Indicates the intended use of the key: • `sig` represents signature. • `enc` represents encryption.
`n`	The modulus for the RSA public key.
`e`	The exponent for the RSA public key.
`alg`	An optional parameter to identify the specific cryptographic algorithm used with the key. **Note** The key size might not always match the cryptographic algorithm used with the key. For example, an `RSA 256` key can be used when the signing algorithm is `RS384` .
`x5c`	The X.509 certificate chain. The first entry in the array is the certificate to use for token verification. The other certificates can be used to verify this first certificate.

The list of JWKs can be maintained and managed by PingFederate, which can create and rotate the keys, or manually by the PingFederate administrator. The supported algorithms are listed in both the OIDC and OAuth Metadata endpoints.

When PingFederate manages the keys, it includes the `alg` (algorithm) claim for the following keys:

- All elliptic curve keys using any supported algorithm , such as `ES256` , `ES384` , and `ES512`
- Three RSA keys, when the signing algorithm is `RS256` (previous, current, pending)

For manually managed keys (static keys), the `alg` claim is included for the following keys:

- All elliptic curve keys using any supported algorithm, such as `ES256` , `ES384` , and `ES512`
- Two RSA keys using `RS256` (previous, current)

PingFederate traditionally uses one RSA-based key for multiple RSA-based signing algorithms (`RS384` and `RS512`) irrespective of the key size, and when the signing algorithm is `RS256` , the signing key is a dedicated RSA key with the `alg` header `RS256` . Not all RSA-based keys support the `alg` header.

The following is an example of the JWKS response with an array of two JWK objects:

```
    {
      "keys": [
         {
            "kty": "RSA",
            "kid": "1",
            "use": "sig",
            "n":
"t7jW8PvJRA7qo4N4dY7JZt1vNtLX9SdRyV1ytW8Fv2jKgGJfRdKjSNWZIvA2fO3efrjzb35LTUpatb0x6OcTjID0J6YwO6UZQL0xfDeX9jK78B2JGpxS
hsC-VzHf2ggn_rBBFBRHvLcZlGSlpjOyh7X9dNIx-rKjyZH8OYdY7db4nxHzvZV7LZjKzRt7S9jkG5Qh7Kol8F1JwYjGIVf03QmUR-
yWZnGGFJbP9oEJMW_kRWVCn_-Ic6wgK1NIDJFjjUZP6oZgQzAjbAULNV1sHsMYyrDTyOAc_26fTJp6MLDeUZ_iO8jKkrrgXsDxUnsztOeHOrGnvMw",
            "e": "AQAB",
          },
          {
             "kty": "RSA",
             "kid": "2",
             "use": "sig",
             "alg": "RS256",
             "n": "uysKtR1Gz9Sd5Q5np5mKk8Q2YwI5RByz1-Z8ry_vWOBzvBaC87Pc3q3a8K1JxFNv-
Wz_s0cYfB2oEgG0jJeb5yo6H5l5TT8g6PYmJ1r14HMeuV7Kj59aGxEV7y4i1HnV7-
eQX07ZuVnZSgmjJNN3qXL_8nwTBNCHrqtkyLGtjK__s1sXljjnoV7lZjzNq3Y3qyTb-7B2wJXTKjyyVyx1epcnoeZVxP4pLVTmHjK4SLjjL20l4gTXi1h
Kj2QHQXbF7ZdDfCn19cdYQ-CK8h6NGKgJU5Z6U5M5mkhA6rMDZrLlRJvyNJzCFg8WuzF41GXwdrC3Aq3Q",
             "e": "AQAB"
           }
      ]
    }
```

Related links

- Manage digital signing certificates and decryption keys

Configuring static signing keys

Determine when to use static and dynamically rotating keys to sign tokens as needed.

About this task

Specify whether PingFederate should use static or dynamically rotating keys to sign self-contained access tokens, ID tokens, JSON web tokens (JWTs) for client authentication, and JWTs for OpenID Connect request objects.

Steps

1. Go to **Security** > **Certificate & Key Management** > **OAuth & OpenID Connect Keys**.

2. Select the **Enable Static Keys** check box to use static keys for OAuth and OpenID Connect.

> **Note**
>
> Clear this check box to let PingFederate generate and rotate keys automatically for OAuth and OpenID Connect.
> The **Enable Static Keys** check box is not selected by default.

Result:

Once selected, the administrative console displays the following fields under the **Signing Keys** heading.

Key Type	Active	Previous	Publish Certificate
EC with P-256 curve	Optional	Optional	Optional
EC with P-384 curve	Optional	Optional	Optional
EC with P-521 curve	Optional	Optional	Optional
RSA	Required	Optional	Optional

3. Follow these steps to complete the configuration under Signing Keys.

    1. For the RSA key type, select an active signing key and optionally a previous signing key.

        > **Note**
        >
        > If you don't find the desired signing key, click **Manage Certificates** to create it.
        > There is no default selection.

        Result:

        The active signing key and the previous signing key (if configured) are published at the PingFederate JSON Web Key (JWK) Set endpoint `/pf/JWKS`.

        > **Note**
        >
        > The `alg` parameter value is only included for the RSA key that handles *RS256*. All EC keys have a corresponding `alg` param. For example: `"crv": "P-256","alg": "ES256"`.

    2. For each applicable EC (elliptic curve) key type, select an active signing key and optionally a previous signing key.

        > **Note**
        >
        > If you don't find the desired signing key, click **Manage Certificates** to create it. Alternatively, complete the configuration, create the desired signing keys later, and then update the configuration afterward.
        > There is no default selection.

        Result:

        The active signing key and the previous signing key (if configured) are published at the PingFederate JWKS endpoint `/pf/JWKS`.

3. **Optional:** For any key type for which you have selected an active signing key (with or without a previous signing key), select the **Publish Certificate** check box to publish the certificates associated with the active signing key and the previous signing key (if configured) at the PingFederate JWKS endpoint `/pf/JWKS`.

> **Tip**
>
> For each applicable signing key, its associated chain of certificates is published as the `x5c` parameter value.

The **Publish Certificate** check boxes are not selected by default.

4. Click **Save**.

Result

> **Important**
>
> When static keys are enabled, PingFederate uses only static signing keys to sign ID tokens for OAuth clients or to sign JWTs for authentication or request objects (or both) for authorization servers; dynamic keys are not used and are not returned by the PingFederate JWKS endpoint `/pf/JWKS`. Signing algorithms associated with EC key types not configured with an active static signing key are hidden.

For existing clients and identity provider (IdP) connections, if you have previously selected a certain signing algorithm associated with an EC key type (for example, **ECDSA using P256 Curve and SHA-256**) without enabling static keys and then subsequently decide to enable static keys without selecting an active signing key for such EC key type (**EC with P-256 curve** in this example), transactions that involve that signing algorithm will fail. When you revisit the configuration, the administrative console displays an error message. Your options are as follows:

OAuth clients

- Click **Save** to update the value of the **ID Token Signing Algorithm** setting to **Default**, which is the equivalent of selecting **RSA using SHA-256** from the list.
- Select a different value from the **ID Token Signing Algorithm** list and save the configuration.
- Ignore the error and click **Cancel** without updating the configuration. Note that runtime errors persist until the configuration issue is resolved.

These options are applicable to individual clients on the **Client** window and the default setting configured for all clients created via the Dynamic Client Registration protocol on the **Client Configuration Defaults** window.

OpenID Connect IdP connections

- Select a different value from the **Authentication Signing Algorithm** list or the **Request Signing Algorithm** list (or both) and save the configuration.
- Ignore the error and click **Cancel** without updating the configuration. Runtime errors persist until the configuration issue is resolved.

These options are applicable to individual OpenID Connect IdP connections on the **OpenID Provider Info** window.

Related links

- Configuring OAuth clients

- Managing client configuration defaults
- OpenID Connect Relying Party support
- OpenID Provider configuration endpoint
- The JSON Web Algorithms (JWA) specification
- The JSON Web Key (JWK) specification

Configuring static decryption keys

You can specify whether PingFederate should use static or dynamically rotating keys to decrypt asymmetrically-encrypted ID tokens.

About this task

> **Note**
>
> When static keys are enabled, you must also select an active signing key for the RSA key type.

Steps

1. Go to **Security > Certificate & Key Management > OAuth & OpenID Connect Keys**.
2. Select the **Enable Static Keys** check box to use static keys for OAuth and OpenID Connect.

   > **Note**
   >
   > Clear this check box to let PingFederate generate and rotate keys automatically for OAuth and OpenID Connect. The **Enable Static Keys** check box is not selected by default.

   Result:

   Once selected, the administrative console displays the following fields under "Decryption Keys".

Key Type	Active	Previous	Publish Certificate
EC with P-256 curve	Optional	Optional	Optional
EC with P-384 curve	Optional	Optional	Optional
EC with P-521 curve	Optional	Optional	Optional
RSA	Optional	Optional	Optional

3. Follow these steps to configure "Decryption Keys".
   1. For each applicable key type, select an active decryption key and optionally a previous decryption key.

> **Note**
>
> If the desired decryption key is not found, click **Manage Certificates** to create it. Alternatively, complete the configuration, create the desired decryption keys later, and then update the configuration afterward. There is no default selection.

Result:

The active decryption key is published at the PingFederate JSON Web Key Set (JWKS) endpoint `/pf/JWKS`.

2. **Optional:** For any key type for which you have selected an active decryption key (with or without a previous decryption key), select the **Publish Certificate** check box to publish the certificates associated with the active decryption key at the PingFederate JWKS endpoint `/pf/JWKS`.

> **Tip**
>
> Each applicable decryption key’s associated chain of certificates is published as the `x5c` parameter value.

The **Publish Certificate** check boxes are not selected by default.

4. Under "Signing Keys", select an active key for the RSA key type.

> **Note**
>
> If the desired key is not found, click **Manage Certificates** to create it. There is no default selection.

Result:

The active signing key is published at the PingFederate JWKS endpoint `/pf/JWKS`.

5. Click **Save**.

Result

> **Important**
>
> When static keys are enabled, PingFederate uses only static decryption keys to decrypt asymmetrically-encrypted ID tokens it receives from OpenID providers. Dynamic keys are not used and are not returned by the PingFederate JWKS endpoint `/pf/JWKS`.

The following snippet illustrates a sample response returned by the PingFederate JWKS endpoint when dynamic keys are used.

```
$ curl -s https://localhost:8031/pf/JWKS |python -m json.tool
{
  "keys": [
    ...
    {
      "kty": "EC",
      "kid": "I-ZbqeLPG2O5qxSf3n8yKmcGbWI",
      "use": "enc",
      "x": "AUSx-2vdfCjU90KohVs1peISnNUeDmGo3m0_x42PucBr-Gd-mHKXQ8EjTeYgLhFB5SYMV5tntKiezayWkUt9Dodc",
      "y": "AIE6vQYcKdOfyQYzENYQ86MIAwSUo4GR_-dn7m2MvRReXkotWOsFT1WKXi_KjamqJIV2AwAUZL-IQj5mew45lSTM",
      "crv": "P-521"
    },
    {
      "kty": "EC",
      "kid": "S2BbNNK9PtG0nA-EhU5BGpZ-OG8",
      "use": "enc",
      "x": "IKXASh9aDPJ1YaeXUww1YZnZ3kum_WLKvZe8xiNW6W8",
      "y": "7_zp2AuY8MY4WEuneHEzV0cqW0buqcmMGVzRANQ0r2I",
      "crv": "P-256"
    },
    {
      "kty": "EC",
      "kid": "t4-jKfmhEHn3mRc-08Oh3WKA2zE",
      "use": "enc",
      "x": "RiQkv_ArGS7Zc8XsXp0VQpEWz9ZUlbLUWA0VbTcUjWIbOByceGhg-tAj6dlFiorq",
      "y": "aHPQlrJPscdcuHtHokyr-70yBo4nUK-BjWrJgisDxnKJQFLP6YK_dfuOpuVYhFJ5",
      "crv": "P-384"
    },
    {
      "kty": "RSA",
      "kid": "tVP7otNKgIWYep8LPBR3wD3tPNE",
      "use": "enc",
      "n":
"hvHfiamhV4wGC9JHppJZjdKG5K3MvhWwo6PBsSQowGOTeILAbzO8Jfmp7nRxuujTE6k83RXNeWUvTwamGqShXvHzGYJlE2gsc0Az_w5xm-
vjoNZD8Cv0Y9C3R4Ckj6dBL70Osk_NfBR7MYmRA6dV0PJ5k4Lt_vQveXMkylD9XuLFP-
gqooMXkB6FCCLqZZAi0voi3WQ7ECzSta3ke9F5VFl7-4zVjRtJHjM9gGEhd5OkaZioqs9xBHeOrwhPbiPTsIA7ve3No5AlGCgZw654s17zr2Ly4q8QZE7
LmM30kRJnu-dpl_dKixFTdQYIBMmIWGUyuB43XYq106z9CWoOcw",
      "e": "AQAB"
    },
    ...
  ]
}
```

When static keys are used, the PingFederate JWKS endpoint `/pf/JWKS` returns only the configured active keys. The following snippet illustrates a sample response returned by the PingFederate JWKS endpoint when an active key was selected for the **EC with P-384 curve** and **EC with P-521** curve key types.

```
$ curl -s https://localhost:8031/pf/JWKS |python -m json.tool
{
  "keys": [
    ...
    {
      "kty": "EC",
      "kid": "7xKkiMb-YpcK2PcrTUoTrYF8EOI",
      "use": "enc",
      "x": "4p_fZluiHS9qLXQi-cqol1LP5nBrFPcXRKQN5yR3Tz51E0xfY9tmOzLqMQwKfDIh",
      "y": "kWh3up-U2mMYOuhzx4Ba7UX0P03EPLr82PdCUG6E3V53Pgnd2QU6ShWu9lH4-ugw",
      "crv": "P-384"
    },
    {
      "kty": "EC",
      "kid": "pE1XwX8Z6QYhAC7mjZ0OCn4DXAk",
      "use": "enc",
      "x": "ATCOsxg6ce437qMVlrqCyHPDE76hC0wP7Wwb7V8heai60LIDDvIJt-evxTOGn7Iolo9PYET8-Bjhu5Zg5MNxOkF-",
      "y": "AdvUA2YD2kn7COLkFIG2vL2k34CMv7VPxsvbgOJBL2exSziMGPw6YJp2eafuHlBom7bkjv3iFy5dTuGB7B28Zc7A",
      "crv": "P-521"
    },
    ...
  ]
}
```

Related links

- OpenID Connect Relying Party support
- The JSON Web Algorithms (JWA) specification
- The JSON Web Key (JWK) specification

Mapping ID token signing keys to virtual issuers

You can create sets of ID token signing keys in PingFederate, and map each set to one or more virtual issuers for OpenID Connect.

Before you begin

Before you map token signing keys to virtual issuers, configure the necessary static signing keys and virtual issuers. For more information, see Configuring static signing keys and Adding virtual issuers for OpenID Connect.

About this task

When minting an ID token, PingFederate signs the ID token with a key from the right key set based on the authorization request, virtual issuers configuration, and token signing keys configuration. Because of these features, you do not need multiple PingFederate environments to support multiple brands, which especially helps if you participate in Open Banking in the UK or have similar requirements.

Steps

1. Go to **Security** > **Certificate & Key Management** > **Token Signing Keys**.

2. Click **Add Key Set**.

3. Enter the key set’s **Name** and optional **Description**. Click **Next**.

4. Select at least one **Issuer**.

5. Select at an **RSA** signing key in the **Active** column.

6. Optional: Select one or more EC (elliptic curve) signing keys in the **Active** column.

7. Optional: Select **Previous** signing keys next to any of the **Active** keys.

8. Optional: Select the **Publish Certificate** check box next to the **Active** signing keys. PingFederate publishes the certificates associated with these active signing keys and previous signing keys (if selected) at the `/pf/JWKS` endpoint.

9. Click **Save**.

Managing certificates from partners

Manage certificates for various connections involving signature verification, encryption, and back-channel authentication to effectively process messages to and from partners.

About this task

You receive certificates from partners for signature verification, encryption, and back-channel authentication. They are managed within connections.

> **Note**
>
> Depending on the use cases, your connection to the partner might not require signature verification, encryption, inbound (SOAP) back-channel authentication by client certificate, or any such combinations. If so, the **Activation & Summary** window does not display the related administrative window.

Signature verification

Specify one or more certificates that PingFederate can use to validate the digital signatures found in inbound messages from your partners.

Steps

1. Select the connection to reach its **Activation & Summary** window.

2. Select **Signature Verification Certificate**.

3. Click **Manage Certificates**.

   > **Note**
   >
   > You can import, export, review, activate, deactivate, and remove certificates for signature verification on the **Certificate Management** window.

Encryption

Specify a certificate that PingFederate uses to encrypt outbound messages before delivering them to your partners.

Steps

1. Select the connection to reach its **Activation & Summary** window.

2. Select **Select XML Encryption Certificate**.

3. Click **Manage Certificates**.

> **Note**
>
> You can import, export, review, activate, and remove certificates for encryption on the **Certificate Management** window.

Back-channel authentication

Specify a certificate that PingFederate uses to authenticate inbound (SOAP) messages from your partners by their client certificates.

Steps

1. Select the connection to reach its **Activation & Summary** window.

2. Select **SSL Verification Certificate**.

3. Click **Manage Certificates**.

> **Note**
>
> You can import, export, review, activate, and remove certificates for back-channel authentication on the **Certificate Management** window.

Configuring certificate revocation

You can choose whether to use certificate revocation list (CRL) checking or Online Certificate Status Protocol (OCSP) checking as your preferred verification method.

About this task

By default, OCSP revocation checking is enabled. Optionally, on **Security > Certificate & Key Management > Certificate Revocation Checking**, you can enable CRL checking as a backup method for failover. Alternatively, depending on your requirements, you can disable OCSP checking and rely only on CRL checking. For more information, see Certificate validation.

Steps

1. **Optional:** Configure OCSP.

   For more information about each field, see the following table.

Field	Description
Enable OCSP	Turns on OCSP certificate-revocation checking. OCSP checking is enabled by default.
Default OCSP Responder URL	A URL to use for certificate-revocation checking, a backup used only if the OCSP Responder URL is not contained in the certificate.

Field	Description
Default OCSP Responder Signature Verification Certificate	Certificate used to verify that the returned certificate status was sent from the Default OCSP Responder—required if the certificate is not included in the response. Click **Manage Certificates** to import the verification certificate.
Do NOT allow Responder to use cached responses	When not selected, the OCSP Responder uses cached responses when available for the subject certificate for an indicated period of time—see the description for **Next Update Grace Period**. If checked, PingFederate sends a nonce in the request to the Responder, effectively requiring that the status of the certificate be determined in real time. This option is intended to enhance the prevention of Internet replay attacks (in addition to timestamping), where required. **Important** Making this selection might slow down OCSP response time for a request and will increase general processing overhead at the Responder site. This check box is not selected by default.
This Update Grace Period (min)	To consider the response valid, the PingFederate server-clock time must correspond to the `<thisUpdate>` timestamp in the OCSP response, plus or minus the number of minutes set for this field to compensate for clock variances. The default value is `5` minutes.
Next Update Grace Period (min)	If the response includes a `<nextUpdate>` timestamp indicating when updated certificate statuses are available, then PingFederate checks to ensure that the timestamp is not earlier than the current server time, adding this grace period to compensate for clock variances. The default value is `5` minutes.
Responder Timeout (sec)	The allowable response time before the OCSP Responder URL is considered unavailable and processing continues. See the **OCSP Responder is Unavailable** setting. The default value is `5` seconds.
Response Caching Interval (hrs)	The number of hours that PingFederate caches the OCSP response. The default value is `48` hours.
Certificate is Unknown	The certificate does not fall under the purview of the certificate authority (CA) associated with the OCSP Responder. The choices indicate whether an unknown certificate is considered valid, or whether to try CRL checking. The default selection is **Treat as Revoked**.
OCSP Responder is Unavailable	Indicates what action to take if you cannot reach the Responder. The choices indicate whether an unknown certificate is considered valid, or whether to try CRL checking. The default selection is **Treat as Valid**.
OCSP Responder Returns Error	Indicates what action to take if the Responder returns an error. The choices indicate whether an unknown certificate is considered valid, or whether to try CRL checking. The default selection is **Treat as Revoked**.

2. **Optional:** Configure CRL checking.

   For more information about each field, see the following table.

Field/Selection	Description
Enable CRL Checking	Enables CRL revocation checking. ⓘ **Note** CRL checking must remain enabled if any selections for OCSP Error Handling include failover. If OCSP is enabled and no CRL failover is specified, then this selection has no effect. CRL revocation checking is disabled by default.
Treat Unretrievable CRLs as Revoked	If selected, PingFederate immediately aborts the processing associated with the certificate. If not selected, the server treats the certificate as valid but continues trying to retrieve the CRL. This check box is not selected by default.
Next Retry on Resolution Failure (min)	Specifies the number of minutes the server waits before trying to retrieve a CRL when the previous attempt failed—applies only when **Treat Unretrievable CRLs as Revoked** is unchecked. The default value is `1440` minutes, which is 24 hours.
Next Retry on Next Update Expiration (min)	How long the server waits before requesting a new CRL when the most recently retrieved CRL (in cache) has a next-update time in the past. ⓘ **Note** Certain actions in the administrative console, such as saving changes to an identity provider (IdP) adapter instance, reset the CRL cache. When this happens, PingFederate requests new CRLs for subsequent transactions as needed. The default value is `60` minutes.
Verify CRL Signature	When selected (recommended), PingFederate verifies the CRL signature using the public key of the issuer, which must be in the certificate chain or in the list of Trusted CAs. This check box is selected by default.
Proxy Settings	If CRL checking is routed through a proxy server, specify the server's host DNS name or IP address and the port number. The same proxy information applies to OCSP checking, when enabled.

Transitioning to an HSM

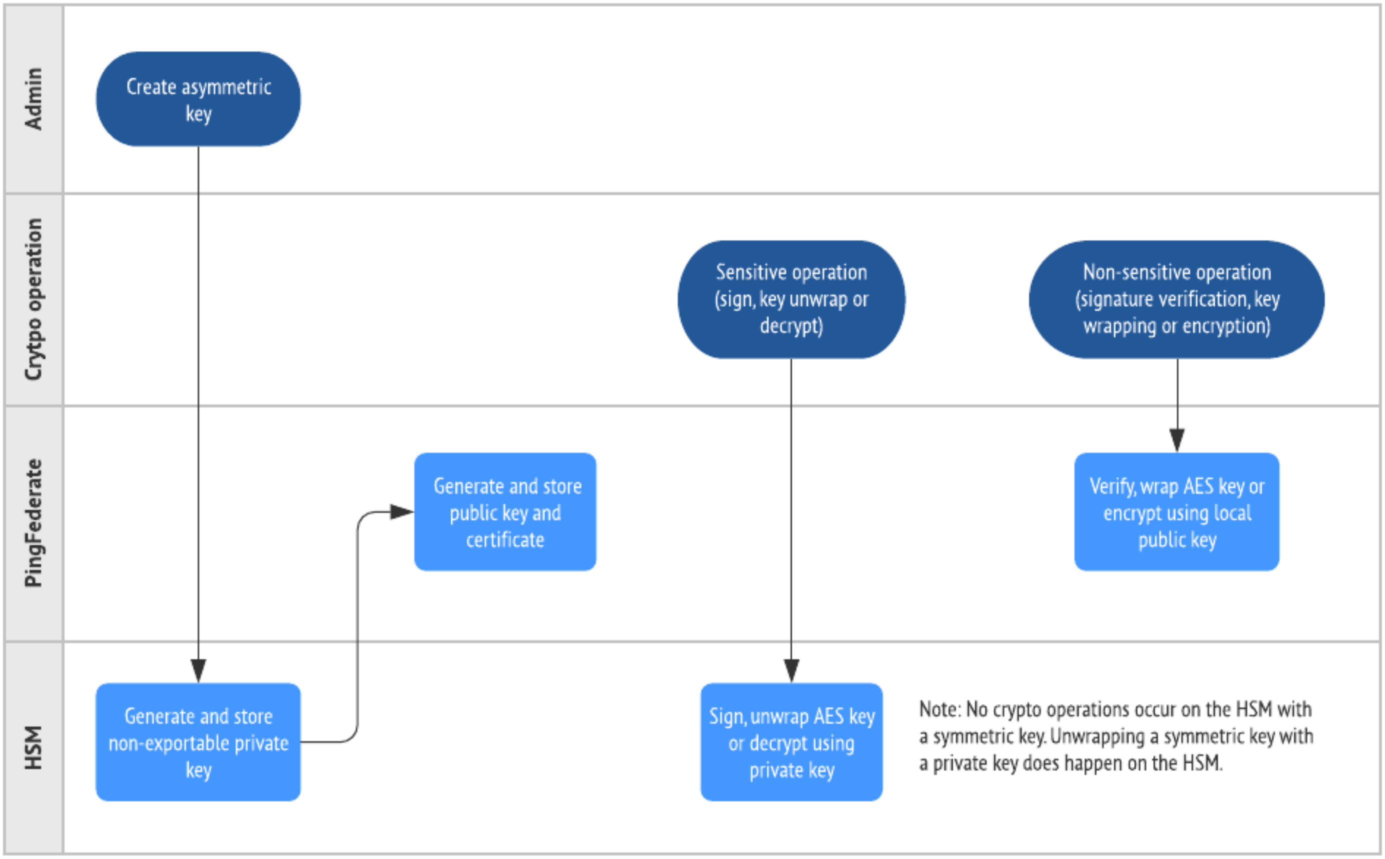
Use the PingFederate administrator functionality to determine whether to store keys and certificates on a hardware security module (HSM) or a local trust store.

About this task

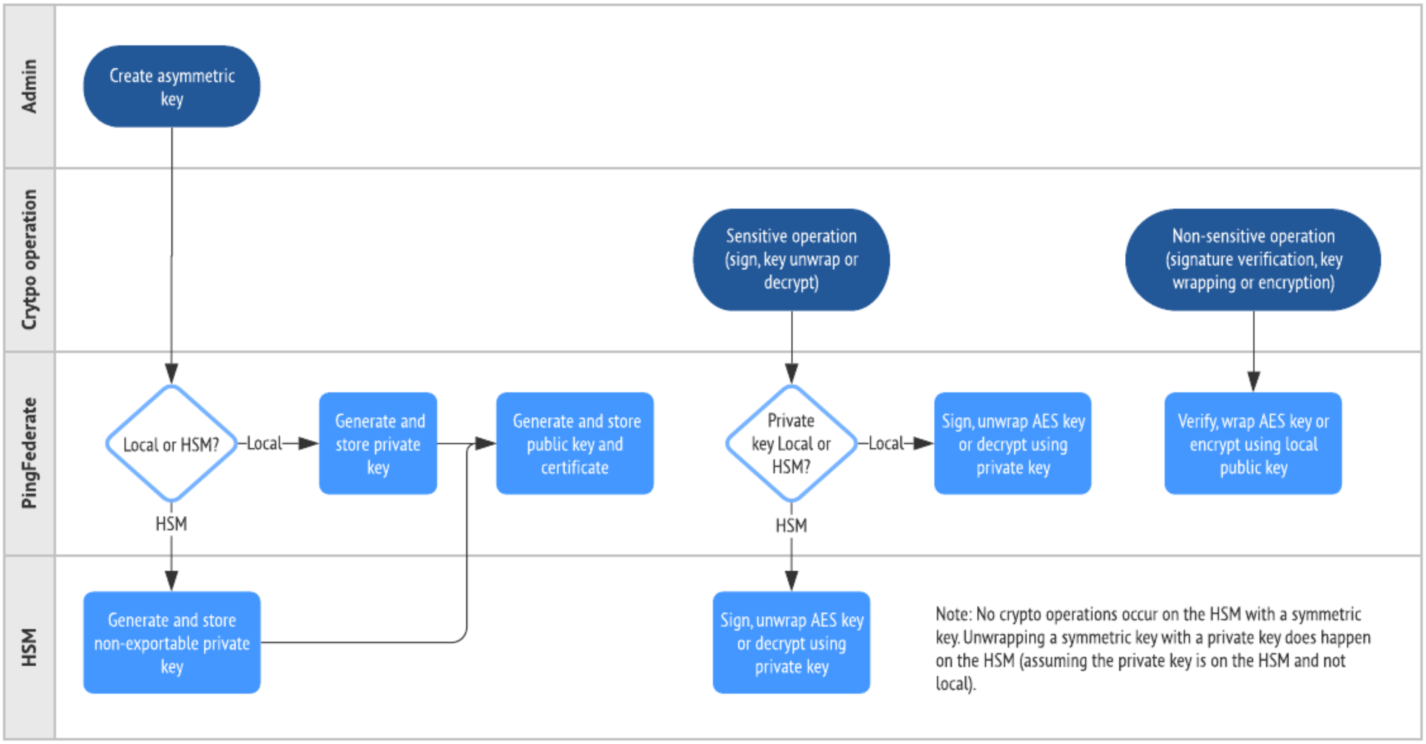
Administrators can enable the HSM hybrid mode, which provides the choice to store each relevant key and certificate on an HSM or the PingFederate-managed local trust store. This capability allows organizations to transition the storage of keys and certificates to a supported HSM to meet security requirements without the need to deploy a new PingFederate environment and mirror the setup.

The following images illustrate some general interactions between PingFederate and an HSM. Those interactions depend on whether you configure the HSM in hybrid mode.

PingFederate not in HSM hybrid mode


Admin
Crytpo operation
PingFederate
HSM
Create asymmetric key
Sensitive operation (sign, key unwrap or decrypt)
Non-sensitive operation (signature verification, key wrapping or encryption)
Generate and store public key and certificate
Verify, wrap AES key or encrypt using local public key
Generate and store non-exportable private key
Sign, unwrap AES key or decrypt using private key
Note: No crypto operations occur on the HSM with a symmetric key. Unwrapping a symmetric key with a private key does happen on the HSM.

PingFederate in HSM hybrid mode


Admin
Crytpo operation
PingFederate
HSM
Create asymmetric key
Local or HSM?
Local
HSM
Generate and store private key
Generate and store public key and certificate
Generate and store non-exportable private key
Sensitive operation (sign, key unwrap or decrypt)
Private key Local or HSM?
Local
HSM
Sign, unwrap AES key or decrypt using private key
Sign, unwrap AES key or decrypt using private key
Non-sensitive operation (signature verification, key wrapping or encryption)
Verify, wrap AES key or encrypt using local public key
Note: No crypto operations occur on the HSM with a symmetric key. Unwrapping a symmetric key with a private key does happen on the HSM (assuming the private key is on the HSM and not local).

Note

For a list of supported HSMs, see the "Hardware security modules" section under "Third-party cryptographic solutions" in System requirements.

When all relevant keys and certificates are stored on the HSM, administrators can turn off the HSM hybrid mode. When the HSM hybrid mode is disabled, PingFederate delegates the management of the relevant keys and certificates to the HSM.

Important

After the HSM hybrid mode is disabled, for keys and certificates that should be stored on an HSM, PingFederate will only access those keys and certificates from the HSM, regardless of whether such keys and certificates exist on the local trust store.

Steps

1. Install and configure the HSM client and the existing PingFederate environment. See Supported hardware security modules.

   ## Important

   When editing the `<pf_install>/pingfederate/bin/run.properties` file, set the `pf.hsm.hybrid` property to `true` to enable the HSM hybrid mode.

   Result:

After PingFederate is integrated with your HSM, you can create (and store) new certificates on your HSM. Because the HSM hybrid mode is enabled, you can reconfigure connections or other configuration items to use the new certificates over a period of time. As long as the HSM hybrid mode is enabled, PingFederate can use certificates stored on your HSM and the local trust store.

> **Important**
>
> When making changes to keys and certificates, you might need to coordinate with your partners. For more information, see Digital signing policy coordination.

2. Create a new SSL server certificate on your HSM and activate it for the administrative console and the runtime server on **Security** > **Certificate & Key Management** > **SSL Server Certificates**.

> **Note**
>
> You can also create separate certificates on your HSM and activate one certificate for the administrative console and the other certificate for the runtime server. For configuration steps, see Manage SSL server certificates.

3. Create new digital signing certificates and decryption keys on **Security** > **Certificate & Key Management** > **Signing & Decryption Keys & Certificates** and reconfigure connections or configuration items to use the new certificates and keys from your HSM.

> **Tip**
>
> Use **Check Usage** to locate the applicable connections or configuration items.

For configuration steps, see Manage digital signing certificates and decryption keys.

4. If your connections support outbound (SOAP) back-channel authentication by client certificates, create new SSL client certificates on **Security** > **Certificate & Key Management** > **SSL Client Keys & Certificates** and reconfigure connections to use the new certificates from your HSM.

> **Tip**
>
> Use **Check Usage** to locate the applicable connections or configuration items.

For configuration steps, see Manage SSL client keys and certificates.

5. If you are transitioning to an Entrust HSM, export the trusted certificate authority (CA) certificates from the local trust store and import them to your HSM on **Security** > **Certificate & Key Management** > **Trusted CAs** and reconfigure configuration items to use the new certificates and keys from your HSM.

> **Tip**
>
> Use **Check Usage** to locate the applicable configuration items.

For configuration steps, see Manage trusted certificate authorities.

6. If you are transitioning to an Entrust HSM, for connections using the unanchored trust model, export the partner certificate for back-channel authentication from the local trust store, import them to your HSM, and reconfigure the connections to use the new certificates from your HSM. For information about the unanchored trust model, see "Trust models" under Digital signing policy coordination.

For configuration steps, see Managing certificates from partners.

Manage Partner metadata URLs

On the **Security > Certificate & Key Management > Partner Metadata URLs** window, you can add, update, review, or remove SAML metadata URLs provided by your partners.

SAML metadata URLs streamline the process of establishing and maintaining SAML connections. If your partner provides SAML metadata by URL, you can use the metadata URL for the following scenarios:

- Creating a new SAML connection using the metadata URL and associating the metadata URL with the new connection
- Enabling or disabling automatic updates from the associated metadata URL
- Adding or updating the metadata URL associated with an existing SAML connection
- Updating an existing SAML connection using the metadata URL instantly

> **Tip**
>
> You can quickly create connections with InCommon participants, update the connections automatically or manually as the InCommon participants update their metadata, and do so securely knowing PingFederate only commits changes to your connections after validating the digital signatures of the signed metadata.

When PingFederate accesses a digitally signed metadata URL for the first time, it validates the digital signature and stores the metadata URL and its verification certificate if the signature is correct. When an existing metadata URL is accessed, PingFederate validates the digital signature using the stored certificate. If there is a digital signature error, PingFederate aborts the process and provides an error with a recommended course of action. You can bypass the signature verification process.

Adding a new metadata URL

Use the **Partner Metadata URLs** window's functionality to add a custom-configured metadata URL.

Steps

1. On the **Partner Metadata URLs** window, click **Add New URL**.
2. On the **URL** tab, define the metadata URL.
   1. Configure each field.

Field	Description
Name	A name of the metadata URL.
URL	The metadata URL.
Validate Metadata Signature	Determines whether PingFederate should validate the digital signature of signed metadata. Select the check box to verify digital signatures. Clear the check box to skip the signature verification process. This check box is selected by default.

   2. Click **Load Metadata**.

3. On the **Certificate Summary** tab, review the certificate information.

   > **Note**
   >
   > This is shown and applicable only when the **Validate Metadata Signature** check box on the **URL** tab is selected.

   - If the metadata is not digitally signed (unsigned), click **Verify** to confirm that the unsigned metadata is reachable at the time of the configuration.
   - If the metadata is signed but the certificate is provided outside of the metadata, click **Import** to upload the verification certificate.

4. On the **Summary** tab, review the configuration. Click **Done** and **Save**.

Updating an existing metadata URL

Use the **Partner Metadata URLs** window's functionality to update and correct the configuration of existing metadata.

Steps

1. On the **Partner Metadata URLs** window, select the applicable metadata by its name.

2. On the **URL** tab, update the name, URL, or digital signature verification option. Click **Next**.

3. On the **Certificate Summary** tab, click **Verify** to confirm that the unsigned metadata is reachable at the time of the configuration or update the verification certificate of a signed metadata.

   > **Note**
   >
   > This is shown and applicable only when the **Validate Metadata Signature** check box on the **URL** tab is selected.

4. If the metadata is signed but the certificate is provided outside of the metadata, click **Import** to upload the verification certificate. Click **Next**.

5. On the **Summary** tab, review the configuration, then click **Done** and **Save**.

Reviewing a metadata URL usage

Use the **Partner Metadata URLs** window's functionality to look over a piece of metadata's information.

Steps

1. On the **Partner Metadata URLs** window, select **Check Usage** for the applicable metadata.

   > **Note**
   >
   > The **Check Usage** option is shown and applicable only when the metadata is used by at least one connection.

2. Review the information in the pop-up window.

3. When finished, close the pop-up window.

Removing a metadata URL

Use the **Partner Metadata URLs** window's functionality to delete an unwanted piece of metadata.

Steps

1. On the **Partner Metadata URLs** window, select **Delete** for the applicable metadata.

   > **Note**
   >
   > The **Delete** option is shown and applicable only when the metadata is not used by any connections.

   To cancel the removal request, select **Undelete** for the certificate.

2. Click **Save** to confirm your action.

Rotating system keys

On the **System Keys** window, you can manually rotate your PingFederate system keys to optimize your environment's security.

About this task

System keys are used in cryptographic operations to generate and consume internal tokens. These tokens are leveraged in multiple use cases such as one-time links for self-service password reset and email ownership verification. Periodic rotation ensures optimal security of your environment.

Steps

1. Go to **Security** > **Certificate & Key Management** > **System Keys**.
2. To rotate the system keys, click **Rotate**.
3. Click **Save**.

Result

PingFederate generates a new **Pending** key. The key that was **Pending** becomes the **Current** key. The key that was **Current** becomes the **Previous** key.

Managing configuration encryption keys

PingFederate maintains a set of configuration encryption keys to encrypt and decrypt sensitive information.

These keys encrypt and decrypt sensitive configuration information and runtime data like the following:

- Datastore passwords
- Adapter shared secrets
- Reversible secrets in OAuth clients
- Data encrypted using the `obfuscate.sh/bat` script, such as the cluster authentication password in the `run.properties` file

- User attributes in access grants and persistent sessions

PingFederate stores configuration encryption keys in the `pf.jwk` file in chronological order, starting with the newest key at the top. The `pf.jwk` file is in the `<pf_install>/pingfederate/server/default/data` directory.

```
{"keys":[
{"kty":"oct","kid":"WTIsxFH5gE","k":"FKDByJ13ZHPpi_oLI2_4q_lFgKNi7J1fx8HTyTJINPc","creationDate":1639163017},
{"kty":"oct","kid":"W31mrSXqnH","k":"Keadc5M4cFoLKdWtR2zaEx8P0Dzs-L2U35JNGgLzDHI","creationDate":1639162773}
]}
```

For encryption, PingFederate uses the newest, or primary, key in `pf.jwk`. For decryption, PingFederate tries each key in the file, starting with the primary key, until it succeeds or until there are no more keys to try.

Managing configuration encryption keys involves regularly rotating the keys and re-encrypting sensitive information with new keys. When `pf.jwk` accumulates keys that are no longer needed for decryption, you can delete them. Use the PingFederate administrative console and the configkeymgr utility to manage your configuration encryption keys.

The administrative console lets you rotate the keys. The configkeymgr utility lets you perform the following tasks:

- List all of the configuration encryption keys.
- Rotate the keys, creating a new key and setting it as the primary encryption key.
- Re-encrypt encrypted configuration data using the primary encryption key.
- Delete unused configuration encryption keys.

The utility, located in the `<pf_install>/pingfederate/bin` directory, comes in two variants:

- `configkeymgr.bat` for Windows
- `configkeymgr.sh` for Linux

When managing configuration encryption keys, PingFederate logs events in the `configkeymgr.log`, `admin.log`, or `admin-api.log`, depending on what you used to perform the tasks.

You should protect your configuration encryption keys with AWS KMS or a custom solution based on the PingFederate SDK (the `MasterKeyEncryptor` interface).

Rotating configuration encryption keys

You can use the PingFederate administrative console to rotate configuration encryption keys.

About this task

To maintain security, you should regularly rotate the configuration encryption keys. Rotating keys involves generating a new key and making it the new primary key. PingFederate will use the new primary key to encrypt sensitive information.

To rotate configuration encryption keys:

Steps

1. In the administrative console, go to **Security** > **Certificate & Key Management** > **Configuration Encryption Keys**.
2. Click **Rotate**.

Result:

PingFederate generates a new key, inserts it into the top of the `pf.jwk` file, and displays it at the top of the **Configuration Encryption Keys** window.

Next steps

After you rotate the configuration encryption keys, you should use the configkeymgr utility to re-encrypt information that was encrypted with previous keys.

Re-encrypting sensitive information with configuration encryption keys

You can use the configkeymgr command-line utility to re-encrypt sensitive configuration information and OAuth client secrets.

About this task

You should re-encrypt sensitive information after you rotate the configuration encryption keys.

To re-encrypt sensitive configuration information:

Steps

1. Stop the PingFederate console node.
2. Run the configkeymgr utility on the console node:
   - If PingFederate is running on Windows, open a command prompt, go to `<pf_install>/pingfederate/bin`, and run `configkeymgr.bat`.
   - If PingFederate is running on Linux, open a terminal window, go to `<pf_install>/pingfederate/bin`, and run `configkeymgr.sh`.

     Result:

   The utility displays its usage help.
3. Run the `reencrypt` command.

   The utility offers optional arguments for the `reencrypt` command.

   Example:

   For example, to perform a dry run of the `reencrypt` command in a Linux environment, enter the following command.

   ```
   ./configkeymgr.sh --reencrypt --dry-run
   ```
4. Restart the PingFederate console node.
5. If PingFederate is running in a cluster:
   1. Replicate the configuration to the engine nodes.
   2. Run the configkeymgr utility on the engine nodes to re-encrypt data that is not included in the replication archive, such as sensitive data defined in the `run.properties` file.

> **Note**
>
> You can run the utility on engine nodes without stopping them.

Deleting unused configuration encryption keys

When PingFederate accumulates configuration encryption keys that are no longer needed for decryption, you can delete them.

About this task

Unused keys accumulate in the `pf.jwk` file as keys are rotated. They also accumulate when data archives are imported. The configkeymgr utility lets you delete unused keys.

The utility lets you delete a specific unused key, all unused keys that were created before a specific time, or all unused keys that were created before a specific key was created. If you use the `--before-timestamp <arg>` parameter, where *<arg>* is a timestamp argument, specify the timestamp value in one of the following formats:

- The format on the **Configuration Encryption Keys** window, which looks like `"Tue Dec 14 14:50:20 PST 2021"`
- The format in the administrative API, which looks like `2021-12-14T22:50:20.000Z`
- The Unix timestamp format from the `pf.jwk` file, which looks like `1639522220`

For more information about the `delete` command's parameters, see the utility's usage help.

When you run the `delete` command, the utility first scans the configuration data, looking for in-use keys. If the utility finds any in-use keys within the scope defined by the command's parameters, it will not delete any keys.

> **Important**
>
> The scan to determine if the keys are in use only works for keys used in PingFederate's configuration or in OAuth clients. The utility does not perform the in-use check on sessions and grants. If you delete in-use keys that are referenced by unexpired sessions or grants, users will need to re-authenticate or re-authorize.
> Before you delete configuration encryption keys, you should wait until the sessions and grants that reference older keys expire. To do that, first determine when the configuration data was last re-encrypted with a new key. Next, find the lifetime of the sessions and grants. Then wait the lifetime of the sessions or grants, whichever is longer, after the re-encryption date. For example, if you re-encrypted the data on June 1st, sessions last 8 hours, and grants last 30 days, then wait at least 30 days after June 1st before you delete the encryption keys.

To delete unused configuration encryption keys:

Steps

1. Stop the PingFederate console node.
2. Run the configkeymgr utility on the console node:
   - If PingFederate is running on Windows, open a command prompt, go to `<pf_install>/pingfederate/bin`, and run `configkeymgr.bat`.
   - If PingFederate is running on Linux, open a terminal window, go to `<pf_install>/pingfederate/bin`, and run `configkeymgr.sh`.

   Result:

The utility displays its usage help.

3. Run the `delete` command with one of the following scope parameters:

    - `--before-timestamp <arg>` deletes any keys that were created before the specified time.
    - `--keyid <arg>` deletes the key with the specified key ID.
    - `--before-keyid <arg>` deletes any keys that were created before the specified key.

    Example:

    For example, in a Linux environment, run the following command to delete unused keys that were created before a specific time.

    ```
    ./configkeymgr.sh --delete --before-timestamp "Tue Jun 15 14:00:00 PST 2021"
    ```

4. Restart the PingFederate console node.
5. If PingFederate is running in a cluster, replicate the configuration to the engine nodes.

System integration

You can configure PingFederate to act in accordance with your circumstances.

If PingFederate acts in a service provider (SP) role, you can configure redirect validation rules to ensure valuable information, such as user attribute values, is sent only to a list of designated target resources. If PingFederate is deployed behind a reverse proxy or a load balancer, you can configure whether and how PingFederate should extract contextual information from the requests. You can manage the availability and authentication requirements for the supporting services that PingFederate offers. You can find these menu items in the **Security** tab under the **System Integration** section.

Configuring redirect validation

Ensure that a designated target exists by validating single sign-on (SSO), single logout (SLO) and self-service user account management transactions.

About this task

You can configure several service provider (SP) adapters to pass security tokens or other user credentials from the PingFederate SP server to the target resource via HTTP query parameters, cookies, or POST transmittal. In all cases, these transport methods carry the risk that a third party (with specific knowledge of the identity provider (IdP), the SP, or both, PingFederate endpoints, and PingFederate configuration) could obtain and use valid security tokens to gain improper access to the target resource.

This potential security threat involves using a well-formed SSO or SLO link to start an SSO or SLO request for a resource at the SP site. However, the target resource designated in the link intercepts the security token by a redirection to a malicious website. This same threat also applies to self-service user account management endpoints when such requests include the `TargetResource` parameter.

To prevent such an attack, PingFederate provides a means of validating SSO, SLO, and self-service user account management transactions to ensure that the designated target resource exists through a list of configurable URLs. At minimum, an expected resource requires a domain name (or an IP address) and the selection of one or more applicable request types.

Note

The following default target URLs are always allowed, and you don't need to enter them into the list manually:

- The default target URL for any IdP connections (see Configuring default target URLs)
- The default target URL for any adapter-to-adapter mappings (see Configuring a default target URL (optional))
- The SP default URL for successful SSO (see Configuring default URLs)
- The IdP default URL for successful SLO (see Configuring a default URL and error message)

PingFederate can also validate the error resource parameter. For more information about the `InErrorResource` parameter, see IdP endpoints, SP endpoints and System-services endpoints.

Important

PingFederate enables both target resource validation and error resource validation by default in new installations. For backward compatibility, PingFederate upgrade tools do not enable these options if they aren't selected in the previous PingFederate installation. Although optional, we strongly recommend enabling validation for both target and error resources and entering all expected resources (including the HTTPS option) to prevent unauthorized access.

Steps

1. Go to **Security > System Integration > Redirect Validation**.
2. Configure target resource validation options.

Option	Description
SSO	When selected, PingFederate validates the requested target resource for IdP connections, adapter-to-adapter mappings, and SAML 2.0 IdP Discovery against a list of configurable resources. This check box is selected by default in new installations. Clear the check box to disable the feature.
SLO and Other	When selected, PingFederate validates the requested target resource for SLO and self-service user account management requests against a list of configurable resources. This check box is selected by default in new installations. Clear the check box to disable the feature.

3. Configure error resource validation.

   Note

   Select the **Enable InErrorResource Validation** check box to validate the requested `InErrorResource` parameter value against a list of configurable resources.
   This check box is selected by default in new installations.
   Clear the check box to disable the feature.

4. Define a list of expected resources.

   Indicate whether to mandate secure connections when this resource is requested under **Require HTTPS**.

> **Important**
>
> This selection is recommended to ensure that the validation will always prevent message interception for this type of potential attack, under all conceivable permutations.

Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

Enter a value without the protocol, such as `example.com` or `10.10.10.10`.

Prefix a domain name with a wildcard followed by a period to include subdomains using one entry. For instance, `*.example.com` covers `hr.example.com` or `email.example.com` but not `example.com`, the parent domain.

> **Important**
>
> While using an initial wildcard provides the convenience of allowing multiple subdomains using one entry, consider adding individual subdomains to limit the redirection to a list of known hosts.

Enter the exact path of this resource under **Valid Path**.

Start with a forward slash, without any wildcard characters in the path. If left blank, any path under the specified domain or IP address is allowed. This value is case-sensitive. For instance, `/inbound/Consumer.jsp` allows `/inbound/Consumer.jsp` but rejects `/inbound/consumer.jsp`.

Select the check box under **Allow Any Query/Fragment** to allow any query parameters or fragment for this resource.

Selecting this check box also means that no query parameter and fragment are allowed in the path defined under **Valid Path**.

This check box is not selected by default.

1. Select one or more request types for this resource.

    - Select the check box under **TargetResource for SSO** if this is an expected SSO target resource for one or more IdP connections, adapter-to-adapter mappings, or SAML 2.0 IdP Discovery.

    - Select the check box under **TargetResource for SLO and Other** if this is an expected target resource for SLO and self-service user account management requests.

    - Select the check box under **InErrorResource** if this is an expected `InErrorResource` parameter value.

      These check boxes are not selected by default.

      Click **Add**.

      Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Use the **Delete** and **Undelete** workflow to remove an existing entry or cancel the removal request.

   Repeat these steps to define multiple expected resources. NOTE: The display order does not matter. A more specific match is considered a better match and an exact match is considered the best match.

5. Click **Save**.

Related links

- Defining a service URL (WS-Federation)

Managing partner redirect validation

PingFederate enables you to validate a parameter for single logout (SLO) to prevent unauthorized access.

About this task

Some of the parameters used to perform redirection represent locations at a partner site—for example, the `wreply` parameter in WS-Federation. To protect against session token hijacking through open redirections, PingFederate provides an option to validate `wreply` for single logout (SLO). Once enabled, the parameter value is managed within the connection on a per-partner basis. PingFederate amalgamates the entries from all active WS-Federation connections and validates `wreply` against the consolidated list.

> **Important**
>
> PingFederate enables `wreply` validation for SLO by default in new installations.
> For backward compatibility, PingFederate upgrade tools do not enable this option if it was not selected in the previous PingFederate installation. Although optional, enabling `wreply` validation for SLO and specifying the allowed domains and paths for each WS-Federation connection can prevent unauthorized access.

Steps

1. Go to **Security** > **Redirect Validation** > **Partner Redirect Validation**.
2. Select the **Enable wreply Validation For SLO** check box to enable this feature.

   > **Note**
   >
   > This check box is selected by default in new installations. Clear the check box to disable the feature.

3. Click **Save**.

Related links

- Defining a service URL (WS-Federation)
- Specifying a service URL (WS-Federation)

Configuring incoming proxy settings

Use the options in the **Incoming Proxy Settings** window to let PingFederate access the information it needs to construct correct responses for incoming requests.

When PingFederate is deployed behind an application load balancer (ALB), a network load balancer (NLB), a reverse proxy, or a similar network traffic management solution on-premises or in the cloud, the following options enable PingFederate to use information in HTTP headers added by the reverse proxy to construct correct responses. These options, configurable on **Security > System Integration > Incoming Proxy Settings**, apply globally to all incoming requests.

HTTP header for client IP addresses

The **HTTP Header for Client IP Addresses** field allows you to globally specify the header name (for example, `X-Forwarded-For`) where PingFederate should attempt to retrieve the client IP address in all HTTP requests sent to PingFederate. Defining this field helps PingFederate identify the correct client IP address when PingFederate is operating behind a load balancer or something similar.

> **Note**
>
> By design, the `X-Forwarded-For` header exposes privacy-sensitive information, such as the IP address of the client. This header is untrustworthy when no trusted reverse proxy exists between the client and server. PingFederate assumes that a trusted proxy verifies any headers configured in the **Incoming Proxy Settings** rather than coming directly from the client. Therefore, we recommend that customers only configure headers in **Incoming Proxy Settings** that are proxied through a trusted source.

Load balancers and proxies commonly append the IP address from an incoming request to the `X-Forwarded-For` (or similar) header. If you enter `X-Forwarded-For` as the value of the **HTTP Header for Client IP Addresses** field, PingFederate combines multiple comma-separated header values in the same order that they are received. Define which IP address you want to use in the list box:

- Leave the default of **Use Last Value** to use the last value in the combined list.
- Select **Use First Value** to use the first value in the combined list.

HTTP header for hostname

The **HTTP Header for Hostname** field allows you to globally specify the header name (for example, `X-Forwarded-Host`) where PingFederate should attempt to retrieve the hostname and port in all HTTP requests sent to PingFederate. Proxies commonly append the hostname and port from an incoming request to the `X-Forwarded-Host` (or similar) header. If you enter `X-Forwarded-Host` as the value of the **HTTP Header for Hostname** field, PingFederate combines multiple comma-separated header values into the same order that they are received. Define which hostname you want to use in the list box:

- Leave the default of **Use Last Value** to use the last value in the combined list.
- Select **Use First Value** to use the first value in the combined list.

Client certificate header name and chain header name

If you use mutual client certificate authentication and want to use the Apache HTTP Server with `mod_ssl` as the incoming proxy, configure the Apache HTTP Server to pass client certificates as HTTP request headers and enter the header names on the **Incoming Proxy Settings** window.

The following examples shows the Apache HTTP Server configured to pass the client leaf certificate and up to four intermediate certificates as headers.

```
...
SSLOptions +ExportCertData
RequestHeader set LEAF_CERT "%{SSL_CLIENT_CERT}s"
RequestHeader set CHAIN0    "%{SSL_CLIENT_CERT_CHAIN_0}s"
RequestHeader set CHAIN1    "%{SSL_CLIENT_CERT_CHAIN_1}s"
RequestHeader set CHAIN2    "%{SSL_CLIENT_CERT_CHAIN_2}s"
RequestHeader set CHAIN3    "%{SSL_CLIENT_CERT_CHAIN_3}s"
...
```

> **Note**
>
> This configuration snippet is for demonstration purposes only.

To configure PingFederate to consume these HTTP request headers for the purpose of mutual client certificate authentication:

- Enter `LEAF_CERT` as the **Client Certificate Header Name**.
- Enter `CHAIN` as the **Client Certificate Chain Header Name**.

  > **Note**
  >
  > Do not enter the trailing number from the chain header names.

> **Caution**
>
> Because HTTP request headers could potentially be forged, you should only specify a **Client Certificate Header Name** and a **Client Certificate Chain Header Name** if the Apache HTTP Server is immediately in front of your PingFederate environment. The specified values must match the header names used in the Apache HTTP Server configuration, omitting the trailing number from the chain header names.

Incoming proxy terminates HTTPS connections

The **Incoming proxy terminates HTTPS connections** option allows you to globally specify that connections to the reverse proxy are made over HTTPS even when HTTP is used between the reverse proxy and PingFederate.

Configuring service authentication

Administrators with the **Admin** role can activate and configure authentication for Attribute Query, Java Management Extensions (JMX), and SSO Directory Service.

About this task

If you are using the SAML 2.0 Attribute Query profile as a service provider (SP), then the requesting applications at your site must authenticate to the PingFederate server. For more information, see Attribute Query and XASP and the developers_reference_guide:pf_sp_services.adoc#spStartAttributeQueryPing SP application endpoint.

Authentication is required to access PingFederate runtime data via JMX (see Runtime monitoring using JMX) or to make SOAP calls to the Connection Management Service. Authentication is optional for the SSO Directory Service. For more information, see Web service interfaces and APIs and SSO Directory Service.

> **Note**
>
> To help ensure network security, access to all of these services is deactivated when PingFederate is first installed.

To activate and configure authentication for the Connection Management Service, grant the administrators all three administrative roles: **Admin**, **Crypto**, and **User Admin**. For more information, see Connection Management Service.

Steps

- To enable a service:
  1. On **Security > System Integration > Service Authentication**, select **Action > Activate** for your desired service.
  2. Enter or modify) the service account **ID** and define or reset the **Shared Secret**.

     You and the application developer must agree to these values.

  > **Tip**
  >
  > Authentication is optional for the SSO Directory Service.

- To disable a service, on **Security > Service Authentication**, select **Deactivate** under **Action** for your desired service.

  > **Note**
  >
  > Although not accessible when deactivated, the Connection Management Service and the SSO Directory Service are deployed by default with PingFederate. If your organization does not plan to use one or both of these services, you can remove the following WAR file or files:
  >
  > - `<pf_install>/pingfederate/server/deploy2/pf-mgmt-ws.war` for the Connection Management Service
  > - `<pf_install>/pingfederate/server/deploy/pf-ws.war` for the SSO Directory Service

Account lockout protection

Account lockout protection provides a level of security to the user and can operate in multiple ways based on the PingFederate environment.

Account lockout protection prevents user accounts from locking at the underlying user repository based on too many failed authentication attempts. It also adds a layer of protection against brute force and dictionary attacks because the user is locked out for a time period when the number of failed attempts exceeds the threshold. This protection is enabled in many areas of PingFederate, including the HTML Form Adapter, the Username Token Processor, the OAuth resource owner password credentials grant type, and the native authentication scheme for the administrative console and API.

> **Note**
>
> The HTML Form Adapter and the Username Token Processor provide a per-instance setting for the maximum number of failed attempts such that administrators can use unique values for different instances of the adapter or the token processor.

In a PingFederate clustered environment, depending on the chosen runtime state-management architecture, the account locking-state information is shared across a replica set, multiple replica sets, or all nodes in the cluster.

Settings for account lockout protection are stored in the `com.pingidentity.common.security.AccountLockingService.xml` configuration file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Related links

- Account Locking Service
- Adaptive clustering
- Directed clustering

Configuring account lockout protection

Use PingFederate's functionality to customize your account lockout protection settings.

Steps

1. Edit the `com.pingidentity.common.security.AccountLockingService.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   The following table provides more information about properties in the `com.pingidentity.common.security.AccountLockingService.xml` file.

Property	Description
MaxConsecutiveFailures	The maximum number of failed attempts before a user is locked out for a time period. The default value is `3` . **Note** The per-instance setting in the HTML Form Adapter and the Username Token Processor overrides this property.
LockoutPeriod	The amount of time in minutes that a user is locked out when the `MaxConsecutiveFailures` threshold is reached. The default value is `1` minute.

   If you have a PingFederate clustered environment, edit this file on the console node.
2. Save the change.
3. Restart PingFederate.
4. If you have a PingFederate clustered environment, click **Replicate Configuration** in **System > Server > Cluster Management**.

Related links

- Configuring an HTML Form Adapter instance
- Configuring a Username Token Processor instance

Password spraying prevention

Use password spraying prevention to mitigate against attacks which exploit weak or compromised passwords.

Password spraying prevention adds a layer of defense against the attack pattern where bad actors try to gain access to protected resources by using the same password, typically weak or compromised, against multiple accounts from multiple locations. When enabled, PingFederate tracks the number of failed login attempts per password. When the number of failures for a particular password reaches a threshold, that password is temporarily locked out. Password spraying prevention applies to the HTML Form Adapter, the Username Token Processor, and the OAuth 2.0 resource owner password credentials grant type.

While password spraying prevention can help mitigate the risk of unauthorized access, we recommend that you also enforce a good password policy and a multifactor authentication solution, such as PingID, to protect your organization from password spraying attacks.

In a PingFederate clustered environment, depending on the chosen runtime state-management architecture, state information is shared across a replica set, multiple replica sets, or all nodes in the cluster.

Settings for password spraying prevention are stored in the `com.pingidentity.common.security.AccountLockingService.xml` configuration file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Related links

- Account Locking Service
- Adaptive clustering
- Directed clustering

Configuring password spraying prevention

Configure how password spraying prevention functions within your PingFederate environment to customize your login security experience.

Steps

1. Edit the `com.pingidentity.common.security.AccountLockingService.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   For more information, see the inline comments and the following table.

Property	Description
DoPasswordLocking	Enable (`true`) or disable (`false`) password spraying prevention. The default value is `false` .
MaxPasswordAttempts	The maximum number of failed attempts before a password is locked out for a time period. Applicable only if password spraying prevention is enabled. The default value is `5` .

Property	Description
PasswordLockoutPeriod	The amount of time in minutes that a password is locked out when the `MaxPasswordAttempts` threshold is reached. Applicable only if password spraying prevention is enabled. The default value is `5` minutes.

If you have a PingFederate clustered environment, edit this file on the console node.

2. Save the change.

3. Restart PingFederate.

4. If you have a PingFederate clustered environment, click **Replicate Configuration** on **System > Server > Cluster Management**.

Implementing a MasterKeyEncryptor using AWS KMS

During initial startup, PingFederate automatically generates a randomized master key, which by default is not encrypted. If you are running in Amazon Web Services (AWS), you can configure PingFederate to use Amazon Key Management Services (KMS) to encrypt the master key.

Before you begin

- Make sure that you have an active connection to AWS.
- Use AWS KMS to generate a key to use for the PingFederate master key encryption.
- See https://docs.aws.amazon.com/kms/latest/developerguide/overview.html for general information about how you can manage access rights to your keys using key policies or AWS Identity and Access Management (IAM).

About this task

To configure the encryption of the PingFederate master key, modify two files: `service-points.conf` and `com.pingidentity.crypto.jwk.MasterKeySet.xml`.

Steps

1. Stop PingFederate.

2. Open `<pf_install>/pingfederate/server/default/conf/service-points.conf` in a text editor.

3. Locate the following lines near the bottom of the file.

```
master.key.encryptor=com.pingidentity.crypto.jwk.NoOpMasterKeyEncryptor
```

4. To enable master key encryption using AWS KMS, replace the lines shown in step 3 with the following lines.

```
master.key.encryptor=com.pingidentity.pingcommons.aws.key.AwsKmsMasterKeyEncryptor
```

5. Save and close the file.

6. Open `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.crypto.jwk.MasterKeySet.xml` in a text editor.

   The contents of the file are shown here.

```
<?xml version="1.0" encoding="UTF-8"?>
<con:config xmlns:con="http://www.sourceid.org/2004/05/config">
    <!--
        Uncomment the below attribute to use an external key for encryption of PF Master Key.

        <con:item name="keyId"> put the key Id here </con:item>
    -->
    <con:item name="jwkEncrypted">false</con:item>
</con:config>
```

7. Uncomment the `<con:item name="keyId">` attribute and specify the key that you generated using AWS KMS. For example, after you've made the change, the file might look like the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<con:config xmlns:con="http://www.sourceid.org/2004/05/config">
    <con:item name="keyId">b3867a2c-4d15-8e0c-6f7b-0b1e61f7ad36</con:item>
    <con:item name="jwkEncrypted">false</con:item>
</con:config>
```

8. Save and close the file.

9. Start PingFederate.

Result

After configuring and starting PingFederate, the PingFederate master key file, `pf.jwk`, is encrypted.

Self-service user account management

As an administrator, you can enable certain self-service applications to let end users better manage their accounts and, by extension, lower identity management costs.

PingFederate provides various self-service applications for end users to manage their accounts. These optional capabilities lower the costs of identity management by freeing administrators from round-the-clock service requests to change passwords, reset passwords, unlock accounts, and recover usernames. Designed for ease of deployment, these capabilities are integrated into the HTML Form Adapter. Administrators can easily enable some or all capabilities with a few configuration changes on a per-adapter basis. Like other user-facing windows, you can customize and localize the user-facing templates to provide the desired user experience.

PingFederate also allows users to unlock their accounts without submitting a ticket to the IT department. When enabled with SSPR, if an account is locked, a user can initiate an account unlock request at the **Sign On** window or the per-adapter Password Reset endpoint. Through the HTML Form Adapter, PingFederate prompts the user to prove ownership of the account using the password reset flow.

When users succeed in proving account ownership, they are allowed to retain their current passwords or to reset their passwords as needed. Furthermore, self-service account unlock is only compatible with PingDirectory and Microsoft Active Directory. If the underlying datastore is connected to Oracle Unified Directory or Oracle Directory Server, users can only unlock their account by changing their current password through the password reset flow.

> **Tip**
>
> Similarly, when configuring customer identity and access management use cases, administrators can enable end users to manage their local accounts, connect or disconnect one or more social connections, and change or set the password for their local accounts. For more information, see Customer IAM configuration and Enabling profile management.

Related links

- Localizing messages for end users
- Customizable user-facing pages

Configuring self-service password management

In the **IdP Adapters** window, create or modify an instance of the HTML Form Adapter to enable a customized self-service password management capability.

About this task

PingFederate offers self-service username password management for users to change their network password. This optional capability is integrated into the HTML Form Adapter and the Lightweight Directory Access Protocol (LDAP) Username password credential validator (PCV). You can configure PingFederate to generate notification messages when users successfully change the password associated with their accounts through the HTML Form Adapter or when their passwords are about to expire.

If you are validating credentials through the PingOne for Enterprise Directory PCV, you can also enable the change password capability. Notifications for change password and password expiry are not supported at this point.

> **Important**
>
> For self-service password management to work correctly with PingDirectory, you must grant the service account the `password-reset` privilege. In PingDirectory, use the `ldapmodify` command to apply the following change:
>
> ```
> dn: uid=pfadmin,ou=People,dc=example,dc=com
> changetype: modify
> add: ds-privilege-name
> ds-privilege-name: password-reset
> ```

Steps

1. In the PingFederate administrative console, go to **Authentication** > **Integration** > **IdP Adapters**.
2. To create a new HTML Form Adapter instance, click **Create New Instance**. To reuse one, select an existing HTML Form Adapter instance.

   If you are reusing an existing HTML Form Adapter instance, skip to step 4c to configure your adapter instance to enable self-service password management.

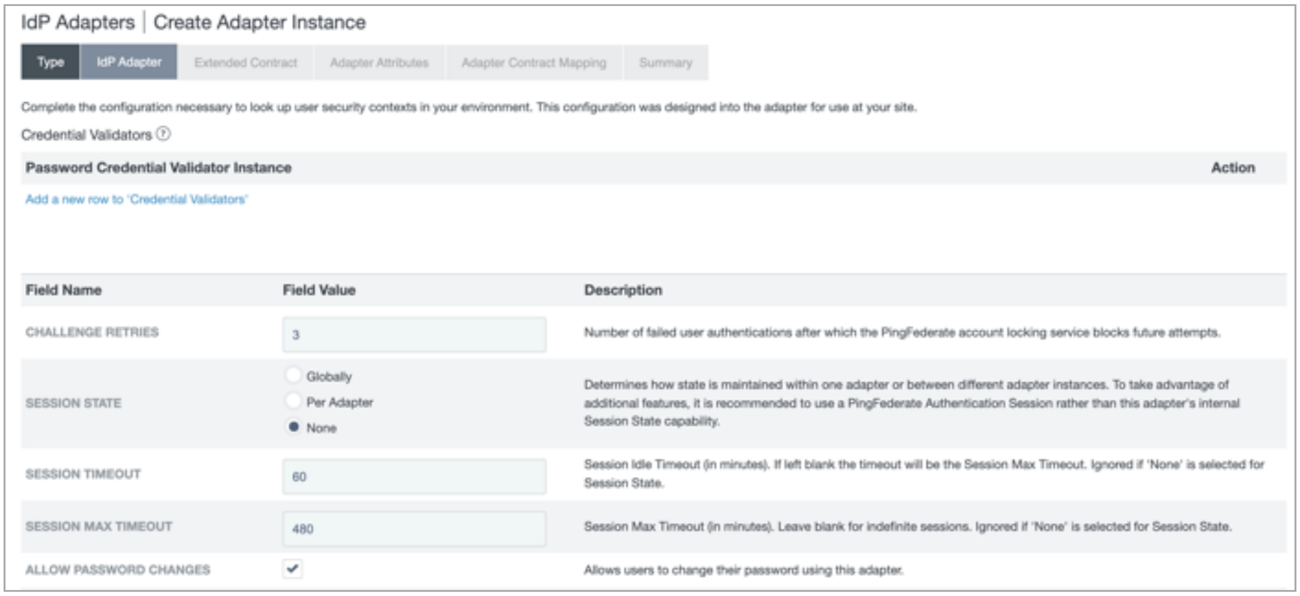
3. On the **Type** tab, configure your adapter instance settings. Click **Next**.

4. On the **IdP Adapter** tab:

    1. In the **Password Credential Validator Instance** section, select the PCV instance as the credential validator.

    2. (Optional) Update any default values or options.

    3. Select the **Allow Password Changes** checkbox.



IdP Adapters | Create Adapter Instance

Complete the configuration necessary to look up user security contexts in your environment. This configuration was designed into the adapter for use at your site.

Credential Validators

Password Credential Validator Instance	Action
Add a new row to 'Credential Validators'	

Field Name	Field Value	Description
CHALLENGE RETRIES	3	Number of failed user authentications after which the PingFederate account locking service blocks future attempts.
SESSION STATE	○ Globally ○ Per Adapter ● None	Determines how state is maintained within one adapter or between different adapter instances. To take advantage of additional features, it is recommended to use a PingFederate Authentication Session rather than this adapter's internal Session State capability.
SESSION TIMEOUT	60	Session Idle Timeout (in minutes). If left blank the timeout will be the Session Max Timeout. Ignored if 'None' is selected for Session State.
SESSION MAX TIMEOUT	480	Session Max Timeout (in minutes). Leave blank for indefinite sessions. Ignored if 'None' is selected for Session State.
ALLOW PASSWORD CHANGES	☑	Allows users to change their password using this adapter.

    4. Configure your adapter instance options. Learn more in the following table.

Option	Effects
Change Password Notification	Select if you want PingFederate to generate a notification message for the user who has successfully changed their password through the HTML Form Adapter. **Note** The message is sent to the user’s email address, specifically the mail attribute value returned by the LDAP Username PCV instance.
Show Password Expiring Warning	Select if you want the **Sign On** page to warn the user about an approaching password expiration.
Change Password Notification	Select to choose a notification publisher instance. **Note** If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

Option	Effects
Show Advanced Fields	Click to review or modify default values related to the change password capability. For example, update the **Change Password Template** field if you want to use a custom template to render the **Change Password** page.

5. (Optional) Customize and localize the on-window messages and notification messages.

Result

You have created a new instance or modified an existing instance of the HTML Form Adapter with the self-service password management capability.

When a user signs on through this adapter instance, the user has the option to change the password associated with the account using the **Change Password** link.

> **Tip**
>
> You can also provide your users the per-adapter Change Password endpoint `/ext/pwdchange/Identify`, which allows them to change their password through this HTML Form Adapter instance without submitting single sign-on (SSO) requests.

Related links

- Customizable user-facing pages
- IdP endpoints
- Managing notification publisher instances
- Configuring an LDAP connection
- Configuring the LDAP Username Password Credential Validator
- Configuring the PingOne for Enterprise Directory Password Credential Validator
- Configuring an HTML Form Adapter instance
- Customizable email notifications
- Localizing messages for end users

Configuring self-service account recovery

PingFederate offers self-service password reset for users to recover their accounts if they forgot their passwords.

About this task

Integrated into the HTML Form Adapter and password credential validator (PCV) framework, users reset their passwords through one of the following mechanisms:

- Authentication policy
- One-time link through email
- One-time password through email
- One-time password through text message
- PingID - The PingID account recovery option requires users to already have a PingID account.

The self-service password reset capability relies on the HTML Form Adapter and the Lightweight Directory Access Protocol (LDAP) Username PCV to query the required attributes for the chosen reset mechanism. PingFederate supports PingDirectory, Microsoft Active Directory, Oracle Unified Directory, and Oracle Directory Server out-of-the-box. Custom PCV implementations can also be developed to offer the self-service password reset features for users stored in non-LDAP data sources. Learn more about the `ResettablePasswordCredential` interface in its Javadoc.

> **Tip**
>
> The Javadoc for PingFederate is located in the `<pf_install>/pingfederate/sdk/doc` directory.

Steps

1. Create a new LDAP datastore. You can find instructions in Configuring an LDAP connection.

   You can also reuse an existing LDAP datastore connection.

   > **Important**
   >
   > - When connecting to an Active Directory (AD) LDAP server, you must secure the datastore connection using LDAPS. AD requires this level of security to allow password changes.
   > - When connecting to PingDirectory, Oracle Unified Directory, or Oracle Directory Server, configure proxied authorization for the service account on the directory server. Learn more in Proxied authorization.
   > - When connecting to PingDirectory, you can configure the `pwdAccountLockedTime` attribute type for the service account on the directory server to allow PingFederate account recovery to unlock locked PingDirectory accounts. Learn more in Allowing PingFederate to unlock PingDirectory accounts.
   > - For self-service account recovery to work correctly with PingDirectory, you must grant the service account the `password-reset` privilege. In PingDirectory use the `ldapmodify` command to apply the following change:
   >
   > ```
   > dn: uid=pfadmin,ou=People,dc=example,dc=com
   > changetype: modify
   > add: ds-privilege-name
   > ds-privilege-name: password-reset
   > ```

2. Create an LDAP username password credential validator. You can find instructions in Configuring the LDAP Username Password Credential Validator.

   The advanced fields on the **Instance Configuration** tab allow you to configure self-service password reset, account unlock, and user name recovery through the HTML Form Adapter

3. Create a new HTML Form Adapter instance. Configuring an HTML Form Adapter instance and HTML Form Adapter advanced fields have complete field descriptions.

    1. Go to **Authentication** > **Integration** > **IdP Adapters**.

    2. In the **IdP Adapters** page, click **Create New Instance**.

    3. On the **Type** tab, enter an instance name and ID.

    4. From the **Type** list, select **HTML Form IdP Adapter**, and click **Next**.

    5. On the **IdP Adapter** tab, click **Add a new row to 'Credential Validators'** and select the LDAP Username PCF instance defined in step 2.

    6. Select the **Allow Password Changes** checkbox.

    7. Select the **Change Password Notification** checkbox if you want PingFederate to generate a notification message for a user who has successfully changed their password through the HTML Form Adapter.

        The message is sent to the user’s email address, specifically the mail attribute value returned by the LDAP Username PCV instance.

    8. Select a **Password Reset Type**. See the following table for more information.

Password Reset Type	Select one of the following methods for self-service password reset. ***Authentication Policy*** Based on the policy contract selected from the **Password Reset Policy Contract** list, PingFederate finds the applicable authentication policy to handle self-service password reset requests. If the users are able to fulfill the authentication requirements as specified by the policy, PingFederate allows the users to reset their password. ***Email One-Time Link*** Users receive a notification with a URL to reset their password.If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. ***Email One-Time Password*** Users receive a notification with a one-time password (OTP) to reset their password.If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**. ***PingID*** Users are prompted to follow the PingID authentication flow to reset their password.Ensure the **PingID Username Attribute** field in the selected LDAP Username PCV instance is configured; otherwise, users will not be able to reset their password.You must also download the settings file from the PingOne admin portal and upload the file to the **PingID Properties** advanced field. **Important** Do not use a method that is already part of a multi-factor authentication (MFA) policy that includes a password challenge as that would indirectly reduce that authentication policy to a single factor. For example, if users normally authenticate with a password challenge and then PingID, the self-service password reset method should not be PingID. Instead, choose the **Authentication Policy** option, select a policy contract from the **Password Reset Policy Contract** list, and configure an authentication policy for self-service password reset. ***Text Message*** Users receive a text message notification with an OTP to reset their password.Ensure the **SMS Attribute** field in the selected LDAP Username PCV instance is configured. Otherwise, users will not receive text message notification for password reset.If you have not yet configured SMS provider settings in PingFederate, click **Manage SMS Provider Settings**. ***None*** Users cannot reset password through this HTML Form Adapter instance. The default selection is **None**. If a notification publisher instance is configured, PingFederate generates a notification for the user who has successfully reset the password through the HTML Form Adapter. The destination is the user's email address, specifically the value of the attribute defined by the **Mail Attribute** setting in the LDAP Username PCV instance.

Field	Description
Password Reset Policy Contract	If you use an authentication policy to handle SSPR requests, you must select a policy contract here. This policy contract doesn't require any extended attributes because PingFederate uses this policy only to find the applicable authentication policies for password resets. **Important** You must use a policy contract dedicated only to password reset. You can't use this policy contract for single sign-on (SSO) anywhere else. To define a policy contract solely for password reset, click **Manage Policy Contracts**. An authentication policy that uses this contract allows users to reset their password. Ensure the policy uses strong authentication methods to securely identify the user who initiated the password reset operation. Map the incoming user ID for adapters in the policy to **Requested User** and confirm that adapters will only return success when this user is the one authenticating. Developing IdP adapters has guidelines on designing adapters for use in password reset or password change authentication policies.

9. Select the **Account Unlock** checkbox if you want to enable self-service account unlock as well.
10. Select a notification publisher instance from the list.

    If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.
11. Click **Show Advanced Fields** to review or modify the rest of the default values related to self-service password reset. HTML Form Adapter advanced fields has descriptions of all advanced fields.

4. If you selected **Authentication Policy** as the password reset type, create a new authentication policy to handle self-service password reset requests.

   Generally a password reset policy must authenticate users through means other than prompting for the forgotten passwords. It should also enforce MFA for added security. Consider the following sample use case.

   You have already created an authentication policy to protect SSO requests. This policy uses an HTML Form Adapter instance to validate user credentials and an instance of the PingID Adapter for MFA. If users satisfy both authentication requirements, the policy uses a policy contract to relay user attributes to partners. Learn more about this policy configuration in Defining authentication policies based on group membership information.

   Like SSO, you also want to protect self-service password reset with MFA.

   Knowing your company actively manages client certificates on company devices, you have decided to use an instance of the X.509 identity provider (IdP) Adapter (named X.509) as the first-factor authentication source in your password reset policy. You've extended the adapter contract with a `CN` attribute, through which the adapter exposes the username found in the client certificate.

For added security, you intend to leverage PingID as the second-factor authentication source. Per step 3e, you have also created a new policy contract (named SSPR APC) for the sole purpose of SSPR. At this point, you are ready to create your password reset policy.

1. On **Authentication** > **Policies** > **Policies**, click **Add Policy**.
2. On the **Policy** window, enter a name and an optional description) for the policy.
3. Select the X.509 IdP Adapter instance.
4. Configure each policy path out of the X.509 Adapter instance.

   Fail

   Select **Done**, which terminates the self-service password reset request. For instance, if a user submits an self-service password reset request from a personal device, the request will fail because the browser on the personal device is not equipped with the company-managed client certificate issued to that user (that is only available on that user's company device).

   Success

   Select the same PingID Adapter instance that you have created and used in the SSO policy.
5. Configure the incoming user ID for the PingID Adapter instance.
   1. Click **Options** to open the **Incoming User ID** dialog.
   2. Select **Adapter (X.509)** under **Source**.
   3. Select **CN** under **Attribute**.
   4. Click **Done** to close the **Incoming User ID** dialog.

      Learn more in Specifying incoming user IDs.
6. Configure each policy path out of the PingID Adapter instance.

   Fail

   Select **Done**, which terminates the self-service password reset request.

   Success

   Select **SSPR APC**, which is the policy contract created solely for password reset per step 3e.

   > **Important**
   >
   > You must not reuse this policy contract for SSO elsewhere.
7. Configure the contract fulfillment for the selected policy contract.

   Because the sole purpose of the selected policy contract is to route the SSPR requests through this password reset policy, the fulfillment of this contract does not matter. It is not used elsewhere. For instance, you can configure its mapping as follows.

Contract Attribute	Source	Value
subject	Text	Benign

8. Click **Done** and **Save**.

   This sample use case demonstrates the capability and flexibility that a password reset policy offers. Depending on actual use cases, you can use a different series of authentication sources to authenticate users in a secure manner. For example, if your organization manages devices using AirWatch, you can add an instance of the AirWatch Adapter as one of the authentication sources in the password reset policy. Other similar solutions include MobileIron and Microsoft Intune.

5. (Optional) Customize and localize the on-screen messages and notification messages.

Result

You successfully created a new instance or modified an existing instance of the HTML Form Adapter with the SSPR and account unlock capabilities.

When a user signs on through this adapter instance, the user has the option to reset the password or unlock the account using the **Trouble Signing On** link.

Additionally, you can also provide your users the per-adapter Account Recovery endpoint `/ext/pwdreset/Identify`, which allows them to reset their password or unlock their account through this HTML Form Adapter instance without submitting SSO requests.

Related links

- Customizable email notifications
- Customizable text message
- IdP endpoints
- Customizable user-facing pages
- Managing notification publisher instances
- Configuring an LDAP connection
- Configuring the LDAP Username Password Credential Validator
- Configuring an HTML Form Adapter instance
- Managing SMS provider settings
- Localizing messages for end users

Configuring self-service user name recovery

Use PingFederate’s self-service user name recovery feature to enable users to recover their lost user names through their email addresses.

About this task

PingFederate offers self-service user name recovery for users to recover their accounts through email if they forget their user names.

This optional capability is integrated into the HTML Form Adapter and the LDAP Username Password Credential Validator (PCV). PingFederate supports PingDirectory, Microsoft Active Directory, Oracle Unified Directory, and Oracle Directory Server out-of-the-box. Custom PCV implementations can also be developed to offer the same capability for users stored in non-LDAP data sources. Learn more about the `RecoverableUsername` interface in its Javadoc.

> **Tip**
>
> The Javadoc for PingFederate is located in the `<pf_install>/pingfederate/sdk/doc` directory.

Steps

1. Go to **Authentication** > **Integration**.On the **IdP Adapters** page, create a new HTML Form Adapter instance.

   You can also reuse an existing HTML Form Adapter instance. If you do, skip to step 1c to configure your adapter instance to enable the self-service user name recovery capability.

   1. Select the HTML Form Adapter instance as the credential validator.
   2. (Optional) Update any default values or options.
   3. Select the **Enable Username Recovery** checkbox.
   4. Select a notification publisher instance from the list.

      > **Note**
      >
      > If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

   5. Click **Show Advanced Fields** to review or modify default values related to self-service user name recovery.

      Example:

      Select the **Require Verified Email** checkbox if you want PingFederate to only send user name recovery email messages to users who have proven ownership of their email addresses.

2. (Optional) Customize and localize the on-page messages and notification messages.

Result

You successfully created a new instance or modified an existing instance of the HTML Form Adapter with the self-service user name recovery capability.

When a user signs on through this adapter instance, the user can optionally recover the user name using the **Trouble Signing On** link.

You can also provide your users the per-adapter username recovery endpoint `/ext/idrecovery/Recover`, which allows them to recover their user name through this HTML Form Adapter instance without submitting single sign-on (SSO) requests.

Related links

- Managing notification publisher instances
- Configuring an LDAP connection
- Configuring the LDAP Username Password Credential Validator
- Configuring an HTML Form Adapter instance
- Customizable email notifications
- Localizing messages for end users
- Customizable user-facing pages
- IdP endpoints

Service provider SSO configuration

You can use the PingFederate administrative console as a service provider (SP) to configure local application-integration information and to manage connections to your identity provider (IdP)-partner sites.

Only one connection is needed per partner, even if integrating more than one web application.

While you define your entity ID on the **Federation Info** tab of the **Protocol Settings** window, you can identify your organization differently through the use of virtual server IDs on a per-connection basis. For more information, see Multiple virtual server IDs.

Additionally, you can deploy an SP connection to bridge a service provider to one or more identity providers through one or more authentication policy contracts. For more information, see Federation hub use cases and Federation hub and authentication policy contracts.

> **Note**
>
> This topic applies to configuration settings needed for browser-based single sign-on (SSO). Although this information also applies to WS-Trust security token service (STS), if you are using PingFederate exclusively as an STS, start with WS-Trust STS configuration.

SP application integration settings

The integration of local applications with PingFederate is the essential "last-mile" configuration that allows end-users at your identity provider (IdP) partner's website to access your protected resources.

The use of application-integration kits and a robust SDK facilitates the integration of local applications with PingFederate. For more information, see Integration overview.

You can configure the service provider (SP) adapters that PingFederate uses to create user sessions that allow single sign-on (SSO) access to your protected resources. You can also set Default URLs to which users can be directed during SSO or single logout (SLO) and look up system endpoints that application developers at your site need to access PingFederate's SSO/SLO services.

Note

If your PingFederate configuration enables the WS-Trust security token service (STS), you can configure plugin token generators on the **Applications** tab in the **Token Exchange > Token Generators** window. For more information, see Service provider STS configuration.

Managing SP adapters

A service provider (SP) adapter creates a local-application session for a user to provide single sign-on (SSO) access to your applications or other protected resources. You must configure at least one instance of an SP adapter to set up connections to identity provider (IdP) partners.

About this task

You can configure multiple instances of adapters, based on one or more adapters, to accommodate the varying needs of your IdP partners.

PingFederate comes bundled with OpenToken Adapter. You can deploy additional integration kits from the Ping Identity Downloads website.

Steps

1. Go to **Applications > Integration > SP Adapters**.
2. In the **SP Adapters** window, choose from the following options.

Option	Description
Configure a new instance	Click **Create New Instance**
Modify an existing instance	Click the name of instance in the **Instance Name** column
View the usage of an existing instance	Click **Check Usage** in the **Action** column on the instance's row
Remove an existing instance	Click **Delete** in the **Action** column on the instance's row

Important

After installing new adapter program files, you might be required to make additional configuration changes in areas such as adapter instances and connections as prompted by the administrative console.

Note

By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System > Server > General Settings**.

Creating an SP adapter instance

The first step in creating an adapter instance is choosing an adapter type.

Steps

1. Go to **Applications** > **Integration** > **SP Adapters**.
2. On the **SP Adapters** window, click **Create New Instance**.
3. On the **Type** tab, configure the basics of this adapter instance:
    1. Enter the **Instance Name** and **Instance ID**.
    2. In the **Type** list, select the adapter type.
    3. Optional: In the **Parent Instance** list, select an existing type.

        If you are creating an instance that is similar to an existing instance, consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

1. Click **Next**.

Configuring an SP adapter instance

You are presented with different configuration parameters depending on the type of adapter that you selected.

Steps

- If this is a child instance, select the override check box to modify the configuration.
- If you are configuring an instance of the OpenToken SP Adapter, see Configuring an OpenToken SP Adapter instance for configuration information.
- If you are configuring an adapter from an integration kit (including any SaaS connector), use the information in Integration overview to configure the adapter instance.

Invoking SP adapter actions

Adapters can perform configuration assistance or validation actions; for example, testing a connection to an LDAP server. Actions may also include generation of parameters that might need to be set manually in a configuration file.

Steps

- On the **Actions** tab, complete any actions required, and click **Next**.

Extending an SP adapter contract

You can configure adapters with an option allowing administrators to add to the attributes required for creating usable sessions.

About this task

This feature might be needed by a legacy application that requires different authentication than other applications under the same enterprise identity-management system.

> **Note**
>
> If this is a child instance, select the **Override** checkbox to modify the configuration.

Steps

1. On the **Extended Contract** tab, enter the name of an attribute and click **Add**.

   Repeat this step as needed to add another attribute.

2. Click **Next**.

Identifying the target application

On the **Target App Info** tab, you can enter the name of the target application and the URL of the application icon.

About this task

The URL is accessible through the IdP Adapter interface, `IdpAuthenticationAdapterV2`, in the PingFederate Java SDK. For more information about the SDK, see SDK Developer's Guide. Both fields are optional.

> **Note**
>
> If this is a child instance, select the **Override** checkbox to modify the configuration.

Steps

1. **Optional:** Enter the application name in the **Application Name** field.

2. **Optional:** Enter the URL to the application icon in the **Application Icon URL** field.

3. Click **Next**.

Reviewing an SP adapter configuration

You can review your SP adapter settings before completing the configuration.

Steps

1.
   - To keep your changes, click **Save**.
   - To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
   - To discard your changes, click **Cancel**.

Configuring target URL mapping

When you have more than one target session defined in an identity provider (IdP) connection, you must map the target URL to its target session.

About this task

When PingFederate receives a single sign-on (SSO) or single logout (SLO) request, it compares the target URL against the configured URLs until a match is found. If a match is not found, the SSO request fails.

> **Important**
>
> For target URL mapping to work correctly, you must configure a target resource entry in the **Security > System Integration > Redirect Validation** settings. If you have not done this, follow the instructions in Configuring redirect validation.

For example, this mapping configuration might be necessary in an IdP-initiated SSO scenario that connects to multiple applications at your site. For transactions initiated at your site, this mapping is required for default situations where the target resource and the adapter instance are not specified in the SSO or SLO request. When this information is provided with the service provider (SP) request, the mapping table is ignored. For more information, see SP services.

When bridging an identity provider to multiple service providers, for each service provider supporting the SAML IdP-initiated SSO profile, map the target URLs to the corresponding SP connection.

> **Tip**
>
> In this scenario, PingFederate is a federation hub for the identity provider and the service providers. For more information, see Federation hub use cases.

Finally, if an IdP connection is associated with one or more SP adapters, authentication policy contracts, or both, you also need to map the target URLs to their respective target session.

You manage target URL mappings on the **Applications > Integration > Target URL Mapping** window. The configuration process involves entering a URL and select a target session for it. See the following table for more information.

The order of mapping is significant in that the first matching mapping, from top to bottom, determines which target session receives the request. For example, if two URLs are mapped in the following order.

URL	Session Target
`http://www.example.com/acct101/`	**OpenToken SP Adapter to an local training app**
`http://www.example.com/*`	**SP connection to SP SaaS**

A target URL of http://www.example.com/acct101/ will be mapped to **OpenToken SP Adapter to an local training app** because the target matches the first mapping in the configuration.

If the order of the mappings is reversed, the same target will be mapped to **SP connection to ACME SaaS** because the first mapping in the new configuration, `http://www.example.com/*`, matches the target URL.

Steps

1. Enter a URL.

   The target URLs that align with your configured target sessions. The URLs instruct the PingFederate SP server to route session-creation processing through an SP adapter instance or an SP connection.

   You can use a wildcard (`*`) to match multiple URLs to the same target session but you can use only one wildcard (`*`) per URL.

   If the target URL in the incoming request is not matched by the first entry in this table, subsequent entries are tried until a match is found.

   > **Note**
   >
   > PingFederate tries the next entry if a target session is not allowed based on restrictions imposed. For more information, see Restricting a target session to certain virtual server IDs.

2. Select a target type from the list.

   You can only select a target type from the list when the IdP role is activated with at least one protocol for browser-based SSO.

   If the IdP role is not activated or is activated without any protocol for browser-based SSO, such as SAML or WS-Federation, the **Target Type** value defaults to **SP Adapter**.

3. Select a target session from the list.

   The available values depends on the chosen the **Target Type** list.

4. Click **Add Mapping**.

5. Repeat these steps to add multiple mappings.

Next steps

Use the up and down arrows to re-arrange the order of the mappings. Click **Edit**, **UpdateCancel** to make or undo a change to a mapping. Click **Delete** and **Undelete** to remove a mapping or cancel the removal request.

Configuring Identity Store Provisioners

PingFederate allows you to create custom identity store provisioners to bridge the inbound system for cross-domain identity management (SCIM) processing of PingFederate to your own user store. For example, you might need to create a custom identity store provisioner that works with an application-specific user database schema.

Using the SDK for PingFederate, you can create and test these custom identity store provisioners. For more information, see the PingFederate SDK Developer's Guide.

To support custom attributes, you must add the schema extension and the custom attributes to the identity provider (IdP) connection. Furthermore, you need to take the expected data structure of the custom attributes into consideration when implementing the `IdentityStoreProvisioner` interface and its methods. In other words, your methods must be able to create, read, update, and delete/deactivate the custom attributes and their sub-attributes if the custom attributes are complex attributes to and from your user store. For more information about custom attributes, complex attributes, and other attribute types, see Defining custom SCIM attributes and SCIM 1.1 Core Schema.

ⓘ **Note**

The identity store provisioner option is active only after you enable **Inbound Provisioning**.

ⓘ **Note**

By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System > Server > General Settings**.

Creating an Identity Store Provisioner instance

On the **Type** tab, you begin creating an instance of an identity store that PingFederate uses to bridge the inbound System for Cross-domain Identity Management (SCIM) processing to an external user store using a custom implementation.

Steps

1. Go to **System > Data & Credential Stores > Identity Store Provisioners**.
2. Click **Create New Instance**.
3. On the **Type** tab, enter the a name and an ID in the **Instance Name** and **Instance ID** fields.
4. From the **Type** list, select a provisioner type.

   Available provisioner types are limited to those that are currently installed on your server.
5. **Optional:** Select a parent instance from the **Parent Instance** list.

   If you are creating an instance that is similar to an existing instance, you might consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.
6. Click **Next**.

Defining the Identity Store Provisioner behavior

Different configuration parameters are available on the **Identity Store Provisioner** tab. These options are controlled by the provisioner plugin software (for more information, see topics about the Identity Store Provisioner interfaces in the SDK Developer’s Guide).

Steps

1. Configure the options as needed.
2. Click **Next**.

Extending the Identity Store Provisioner contract

Identity Store Provisioners can be written with an option allowing administrators to add to the core attributes the plugin instance requires.

About this task

Both the core and extended contract attributes you define must be mapped when you configure Write Users within an inbound provisioning connection.

> **Tip**
>
> To keep your plugin flexible across multiple connections, assuming a one-to-one connection-to-identity store provider setup, you might want to hard code a set of core attributes for all connections to fulfill, and then extend attributes on as needed when a partner connection depends on additional attributes.

> **Note**
>
> If this is a child instance, select the override check box to modify the configuration.

Steps

1. On the **Extended Contract** tab, create and modify attributes as needed.

Option	Action
Add an attribute	Enter the attribute name in the text box and click **Add**.
Modify an attribute name	1. Click **Edit** under **Action** for the attribute. 2. Make the change and click **Update**. **Note** If you change your mind, click **Cancel** under **Action**.
Delete an attribute	Click **Delete** under **Action** for the attribute.

2. Click **Next**.

Extending the Identity Store Provisioner contract for groups

Identity Store Provisioners can be written with an option allowing administrators to add to the core group attributes the plugin instance requires.

About this task

Both the core and extended group attributes that you define must be mapped when you configure Write Groups within an inbound provisioning connection.

Tip

To keep your plugin flexible across multiple connections, assuming a one-to-one connection-to-identity store provider setup, you might want to hard code a set of core attributes for all connections to fulfill, and then extend attributes on as needed when a partner connection depends on additional attributes.

Note

If this is a child instance, select the override check box to modify the configuration.

Steps

1. On the **Extended Group Contract** tab, create and modify attributes as needed.

Option	Action
Add an attribute	Enter the attribute name in the text box and click **Add**.
Modify an attribute name	1. Click **Edit** under **Action** for the attribute. 2. Make the change and click **Update** **Note** If you change your mind,click **Cancel** under **Action**.
Delete an attribute	Click **Delete** under **Action** for the attribute.

2. Click **Next**.

Reviewing the Identity Store Provisioner configuration

On the **Summary** tab, review your Identity Store Provisioner configuration.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Configuring default URLs

As a service provider (SP), you can supply a default URL that the end-user might see when a single sign-on (SSO) request succeeds, that is, a session is created at your site, but the target resource is not available or not specified.

To configure default URLs, go to **Applications > Integration > SP Default URLs**.

> **Note**
>
> You can also specify default target SSO URLs for individual identity provider (IdP) connections, which take precedence over this global setting. For more information, see Configuring default target URLs.

Similarly, you can specify to prompt a default URL indicating a successful single logout (SLO) to the end-user if no other page is designated.

> **Note**
>
> The error message is displayed only when the application calling the start-SSO endpoint does not explicitly provide its own error page URL. The default entry in this field is used to localize the message. For information about how to find and change the default English message and how use the PingFederate localization feature, see Localizing messages for end users. If localization is not needed, you can also specify a message directly in this field to change the default.

Your application or your partner's application can supply these URLs at runtime (see SP services). However, if none are provided, PingFederate uses the default values you enter on this window unless, in the case of SSO, a default is also defined for the connection.

> **Tip**
>
> If no default targets are specified here or at the connection level (for SSO), PingFederate provides built-in landing pages for the user. These web pages are is among the templates you can modify with your own branding or other information. For more information, Customizable user-facing pages.

Viewing SP application endpoints

Web-application developers at your site need to know the application endpoints to initiate transactions through PingFederate.

Go to **Help > SP Application Endpoints** to see a list of endpoints and descriptions applicable to your federation role.

> **Note**
>
> These endpoints are built into PingFederate and cannot be changed.

For specific parameters required or allowed for these endpoints, see SP services and System-services endpoints.

Federation settings

If your identity federation uses the SAML 2.0 X.509 attribute sharing profile (XASP), you might need to identify the identity provider (IdP) connection to which an attribute request applies.

If so, use the **System > Protocol Metadata > Attribute Requester Mapping** window to complete the configuration. For more information, see Attribute Query and XASP and Managing attribute requester mappings.

View endpoints that your federation partners need to know to access your services from the **Help** menu.

Managing attribute requester mappings

If you are using the SAML 2.0 X.509 attribute sharing profile (XASP), applications at your site must supply the subject distinguished name (DN) to identify a user's X.509 authentication certificate.

About this task

Optionally, an application can also supply an issuer DN, which can be used to determine the correct identity provider (IdP) attribute authority to use for a set of users associated with an IdP. For more information, see Attribute Query and XASP.

> **Note**
>
> You must set the `Format` query parameter to a specified value for XASP. For more information, see SP services.

You can map X.509 identifying information to connections and specify a default connection on the **System > Protocol Metadata > Attribute Requester Mapping** window.

At runtime, the issuer DN, if supplied, is evaluated against the entries under **Issuer DN Pattern** in hierarchical order until a match is found. If a match is found, the corresponding IdP connection is selected to issue a response to the attribute query request. If the issuer DN matches no entry or if it is not provided, the subject DN from the request is compared against the entries under **Subject DN Pattern** in a similar manner. If the subject DN matches no entry, then the default IdP connection is used.

You can use a regular expression to match different DNs to the same connection. Only one expression can be used in any single entry. DN values must be entered in all lower-case characters.

Steps

1. Map one or more issuer DNs to SAML 2.0 IdP connections, as needed.
   1. Enter an issuer DN under **Issuer DN Pattern**.
   2. Select an IdP connection under **IdP Connection Name**.
   3. Click **Add**.
   4. Repeat these steps to add more entries.
2. Map one or more subject DNs to SAML 2.0 IdP connections, as needed.
   1. Enter a subject DN under **Subject DN Pattern**.
   2. Select an IdP connection under **IdP Connection Name**.
   3. Click **Add**.
   4. Repeat these steps to add more entries.
3. Select a default IdP connection from the list.

Next steps

You can click **Edit**, **Update**, and **Cancel** to make or undo a change to an entry. Click **Delete** and **Undelete** to remove an entry or cancel the removal request.

Viewing SP protocol endpoints

Your federation partners or security token service (STS) clients need to know the applicable service provider (SP) services endpoints to communicate with your PingFederate server. Configured service endpoints for SAML connections are included in metadata export files.

Go to **Help > SP Endpoints** to see a list of applicable OpenID Connect, SAML, WS-Federation, and WS-Trust STS endpoints. A pop-up window displays only those endpoints related to the federation protocols enabled on the **System > Server > Protocol Settings > Federation Info** tab. These endpoints are built into PingFederate and cannot be changed.

PingFederate provides a favorite icon for all protocol endpoints. For more information, see Customizing the favicon for application and protocol endpoints.

The table below describes each endpoint.

Service	URL and Description
Third Party Initiated Login (OpenID Connect 1.0)	`/sp/init_login.ping` The URL that receives and processes login requests initiated by an OpenID Provider (OP) or another party. This protocol endpoint supports HTTP GET and POST methods and the following parameters: • `iss` : the Issuer Identifier of the OP, to which PingFederate sends the authentication requests. This parameter is always required. * `target_link_url` : the destination of the request after authentication. This parameter is required if no default target URL is specified in the SP configuration or the IdP connection. If specified, the parameter value always overrides the default target URL. * `login_hint` : a hint to the OP about the end user. This parameter is optional. If your use case supports a generic login, you can add the `login_hint` parameter with a default value to the IdP connection on the **OpenID Provider Info** window. Furthermore, you may select the check box under **Application Endpoint Override** so that the application can optionally override the login hint value by including the `login_hint` parameter in the URL. Other parameters if any are not sent to the OP unless they are defined with a default value (or default values) in the IdP connection. For more information, see Configuring request parameters and SSO URLs. For more information about Third Party Initiated Login flow, see the OpenID Connect specification.
Single Logout Service (SAML 2.0)	`/sp/SLO.saml2` The URL that receives and processes logout requests and responses.
Assertion Consumer Service (SAML 2.0)	`/sp/ACS.saml2` A SAML 2.0 implementation that receives and processes assertions from an IdP. The numbers reflect the index value PingFederate uses to handle each binding.

Service	URL and Description
Artifact Resolution Service (SAML 2.0)	`/sp/ARS.ssaml2` The SOAP endpoint that processes artifacts returned from a federation partner to retrieve the referenced XML message on the back channel. See the note at the end of this table.
Assertion Consumer Service (SAML 1.x)	`/sp/acs.saml1` A SAML 1.x implementation URL that receives and processes assertions from an IdP.
Single Sign-on Service (WS-Federation)	`/sp/prp.wsf` The WS-Federation implementation URL that receives and processes security tokens and SLO messages.

WS-Trust STS (two endpoints)	`/sp/sts.wst`
	The SOAP endpoint that receives and processes SAML security-token requests from STS clients (web service providers at the SP site), validating SAML tokens or validating and exchanging SAML tokens based on the configured IdP connection.
	`/pf/sts.wst`
	Initiates direct STS token-to-token exchange and token validation, from an IdP token processor to an SP token generator, when that feature is configured. For more information, see Token translator mappings.
	Note
	If you configure multiple token-generator instances of the same type for the connection or token-to-token mapping, a query parameter, `TokenGeneratorId`, must be added to either of these endpoints. For more information, see Managing token generators.
	See the note at the end of this table.

> **Important**
> If mutual SSL/TLS is used for authentication, you must configure a secondary PingFederate listening partner for use by partners or STS clients for the relevant endpoints such as `.ssaml` and `*.wst`. For more information, see Configuring PingFederate properties.

Virtual server ID support

For SAML connections using multiple virtual server IDs, each virtual server ID has its own set of protocol endpoints. For more information, see Multiple virtual server IDs. You can export a connection metadata for your partner on the **System > Protocol Metadata > Metadata Export** window. For more information, see Exporting connection-specific SAML metadata.

For WS-Federation and SAML connections using multiple virtual server IDs, you can provide your partner the federation metadata endpoint (`/pf/federation_metadata.ping`) with the `PartnerIdpId` and `vsid` parameters. See the table below for an example.

Partner's entity ID	Your virtual server ID	Federation metadata URL
SP	idev1	https://www.example.com/pf/sts_mex.ping?PartnerIdpId=IdP&vsid=idev1

Partner's entity ID	Your virtual server ID	Federation metadata URL
	idev2	https://www.example.com/pf/sts_mex.ping?PartnerIdpId=IdP&vsid=idev2

In this example, the base URL and the runtime port of your PingFederate server are www.example.com and 443, respectively.

The federation metadata endpoint returns information that is specific for a given virtual server ID when the request includes the `vsid` parameter.

For WS-Trust STS, you can provide your partner the STS metadata endpoint (`/pf/sts_mex.ping`) with the `PartnerIdpId` and `vsid` parameters. The STS metadata endpoint returns information that is specific for a given virtual server ID when the STS metadata request includes the `vsid` parameter.

For more information about these metadata endpoints, see System-services endpoints.

The virtual server ID concept does not apply to the `/pf/sts.wst` endpoint because token-to-token exchange does not involves any connections. As needed, you can pass the token-to-token endpoint to your partners as-is.

Managing IdP connections

As a service provider (SP) site, you can manage connection settings to support the exchange of federation-protocol messages, such as OpenID Connect, SAML, WS-Federation, or WS-Trust, with an identity provider (IdP), OAuth client, OpenID Provider (OP), or security token service (STS) client application at your site.

These settings include:

- User attributes that you expect to receive in an SSO token such as a SAML assertion or WS-Trust STS SAML token.
- User attributes the you expect the OP to return in an ID token or through its user information, UserInfo, endpoint on-demand.
- User attributes that may be requested using the SAML Attribute Query profile if that profile is used.
- The protocol, profiles, and bindings of the connection, including detailed security specifications such as the use of back-channel authentication, digital signatures, signature verification, and XML encryption.

To establish a connection, you and your partner must have decided this information in advance. For more information, see Federation planning checklist.

As an SP site, you respond to user requests for single sign-on (SSO) and single logout (SLO) by creating or closing user sessions, respectively, in local applications. You integrate these applications with PingFederate by configuring them with SP adapter instances. Furthermore, in preparation for configuring a new SSO connection, you need to know which adapter instance or authentication policy contract to use. For more information, see Managing target session mappings.

No adapter instance or authentication policy contract is required for a connection that uses only the Attribute Query profile. For more information, see Manage the Attribute Query profile in an IdP connection.

If you intend to pass attribute values to an adapter instance from a local datastore, you must define the datastore during this configuration. If you have not done so already, see Datastores.

Administrative interface

You manage connection settings in the **Authentication > Integration > IdP Connections** window, which organizes the settings into a series of primary tasks. Some primary tasks have one or more levels of sub tasks. Each primary or sub task has its own tab, where you manage one or more settings. You can move to a sibling task using the **Next** or **Previous** button. If you are on a sub task, you can also move to its parent task using the **Done** button.

When creating a new connection, you can save your progress using the **Save Draft** button. Note that not all tabs offer this option. When you reach the **Activation & Summary** tab, you must click **Save** to complete the new connection.

When editing an existing connection, you can make changes and then click **Save** to commit your changes. In order words, you are not required to step through all tabs to reach the **Activation & Summary** tab before you can save your changes.

> **Note**
>
> The **Save** button is available on most tabs. If a tab does not show a **Save** button, click **Next** or **Done** until you reach to a tab where you can use the **Save** button to commit your changes.

Accessing IdP connections

In the **IdP Connections** window, you can create or import a connection, or edit a recently modified connection by clicking on its connection name.

About this task

The **IdP Connections** window displays 20 connections at a time. As needed, use the pagination controls to navigate through the rest of your connections. You can also search connections by their names or connection IDs.

> **Tip**
>
> A connection is included in the search results so long as its name or ID is a partial, case-insensitive match to a search term.

You can sort by connection name, partner connection ID, default virtual server ID, creation date, or last modified timestamps; narrow by protocol and status; and perform various connection-related tasks.

Steps

1. Go to the **Authentication > Integration > IdP Connections**.

Choice	Action
Edit a connection	Select the connection by its name. For the setting you want to change, select the corresponding tab and follow the configuration wizard to complete the task.
Create a connection	Click **Create Connection**, then follow the configuration wizard to create a new connection to your identity provider (IdP) partner.

Choice	Action
Copy a connection	Click **Select action > Copy**, then follow the configuration wizard to create a new connection based on an existing (source) connection. This is most useful if the new connection and the source connection share many common setting values.
Export a connection	Click **Select Action > Export Connection**, then save the XML file as prompted. This is useful in situations where you want to make a backup of a connection prior to making changes to it.
Import a connection	Click **Import Connection**, then follow the on-screen instructions to complete the task. If the connection already exists, you have the option to overwrite the existing connection. ⓘ **Note** Prior to the import, you can modify the XML file to suit your needs. The XML file can also be imported to another PingFederate environment acting in the same federation role (SP) at your site. The source and the target must run the same version of PingFederate.
Export metadata for any SAML browser single sign-on (SSO) connection	Click **Select Action > Export Metadata**, then follow the on-window instructions to complete the task.
Update a SAML browser SSO connection	Click **Select Action > Update with Metadata**, then follow the on-screen instructions to complete the task. You can update a connection via a metadata XML file or a metadata URL. **Important** The update operation might require additional configuration. Review the connection after the update operation.
Toggle the status of a connection	Slide the toggle switch to enable or disable a connection.
Remove a connection	Click **Select Action > Delete**.

Choice	Action
Override the verbosity of runtime transaction logging for all IdP connections	Click **Show Advanced Fields** and the select the desired override option. ***Off*** Select this option and let the per-connection **Logging Mode** configuration determine the amount of information PingFederate records in the runtime transaction log. This is the default selection. ***On*** Select this option, followed by one of the four logging modes, to set the verbosity of runtime transaction logging for all IdP connections. This is most useful when troubleshooting an issue that affects multiple connections.
Turn off automatic multi-connection error checking	Click **Show Advanced Fields** and the select the **Disable Automatic Connection Validation** check box. This check box is not selected by default. Once selected or cleared, the state of this setting is reflected on **Applications > Integration > SP Connections** as well. For more information about this advanced setting and its impact, see Configuring automatic connection validation.
Keep your changes	Click **Save**.
Discard your changes	Click **Cancel**.

Resolving IdP connection errors

About this task

PingFederate automatically validates configured connections before displaying them on the **IdP Connections** window. This validation ensures these connections have not been adversely affected by any subsequent changes in the supporting components, such as an adapter instance or an authentication policy contract.

If errors are found, the administrative console displays a visual cue next to the applicable connections.

Steps

- To resolve the error, select the connection and follow the on-window instructions to modify the configuration, one connection at a time.

Choosing an IdP connection type

You can use the administrative console to choose an identity provider (IdP) connection type.

About this task

You can indicate on the **Connection Type** tab whether the connection to this partner is for browser single sign-on (SSO), WS-Trust security token service (STS), OAuth, SAML, inbound provisioning, or a combination of them.

> **Note**
>
> You can add STS, OAuth, and outbound provisioning support to any existing SSO connection, or vice versa, at any time. However, when OpenID Connect is the chosen protocol for browser SSO, the other types become unavailable.

Select the applicable protocol on the **Connection Type** tab when establishing a new connection.

> **Note**
>
> If your partner's deployment also supports multiple protocols and you intend to communicate using more than one, you must set up a separate connection for each protocol. Each connection must use a unique partner connection ID.

Steps

- On the **Connection Type** tab, indicate the desired type of connection to your partner.

Choice	Action
Configure a connection for secure browser-based SSO	Select the **Browser SSO Profiles** check box and a protocol from the list, if necessary.
Configure an STS connection	Select the **WS-Trust STS** check box and the default token type from the list.
Configure a connection that exchanges SAML assertions or JSON web tokens (JWTs) for access tokens	Select the **OAuth Assertion Grant** check box. **Note** The **OAuth Assertion Grant** option is available only if at least one Access Token Manager instance has been configured on the **Applications > OAuth > Access Token Management** window For more information about these standards, see Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants and JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants.
Configure an inbound provisioning connection	Select the **Inbound Provisioning** check box and choose to support provisioning of users only (**User Support**) or users and groups (**User and Group Support**). For groups, nested group membership, if any, is preserved.

- **Optional:** If your PingFederate license manages connections by groups, you can select a group for this connection.

  This option is not displayed for unrestricted or other types of licenses.

Choosing IdP connection options

On the **Connection Options**tab, shown only for browser-based single sign-on (SSO) connections, you can enable browser-based SSO in conjunction with Just-in-Time (JIT) provisioning. Additionally, you can also choose to map user attributes for persistent grants used by the optional PingFederate OAuth authorization server.

About this task

For SAML 2.0, you can configure the **Attribute Query** profile with or without the browser-based SSO.

Steps

- On the **Connection Options** tab, make the appropriate selections for your configuration.

Choice	Action
Create a connection for browser-based SSO.	Select the **Browser SSO** check box.
Enable JIT provisioning, OAuth attribute mapping, or both.	Select the appropriate check box after selecting the **Browser SSO** check box.
Create a connection to facilitate the SAML 2.0 Attribute Query profile.	Select the **Attribute Query** check box. For more information, see Attribute Query and XASP

Importing IdP metadata

You can use the PingFederate administrative console to import and update identity provider (IdP) metadata.

About this task

If you are using one of the SAML protocols without a connection template, you can expedite the setup by one of the following actions:

- Import a metadata file
- Select a metadata URL

When you select a metadata URL, PingFederate also enables the automatic update option and checks the metadata periodically. If PingFederate detects changes in the partner's signing certificates for digital signature verification, encryption key, or contact information, it updates the connection automatically. For better housekeeping, the update process removes verification certificates from the connection when the partner no longer maintains them in its metadata. In a clustered environment, PingFederate automatically replicates verification certificates and encryption key changes to all engine nodes. Offline engine nodes will also consume these changes as they restart and rejoin the cluster. If you prefer to update the connection manually, you can clear the **Enable Automatic Reloading** check box.

You can configure reload frequency at **System** > **Protocol Metadata** > **Metadata Settings** > **Metadata Lifetime** tab. The default reload frequency is daily.

We recommend you turn on notifications for SAML metadata update events at **System** > **Monitoring & Notifications** > **Runtime Notifications**.

> **Note**
>
> The notification message provides a list of the applicable items if the metadata contains changes that require additional configuration.

After creating the connection, you can add, remove, or change the metadata URL associated with the connection in the **Metadata URL** tab. In addition, you can toggle the **Enable Automatic Reloading** check box for the connection.

> **Tip**
>
> Using a metadata URL with automatic reloading streamlines the configuration process. For example, you can quickly establish a browser SSO connection to an InCommon-participating partner. For more information, see www.incommon.org/participants.

Steps

1. Select from one of the following steps to import or update metadata.

Metadata medium	Steps
Metadata file	1. On the **Import Metadata** tab, select the **File** option. 2. Choose the metadata file, and then click **Next**. **Note** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata verification certificate on the **Import Certificate** tab, and then click **Next**. 3. On the **Metadata Summary** tab, review the signature information to evaluate the authenticity of the metadata.

Metadata medium	Steps
Metadata URL	1. On the **Import Metadata** tab, select the **URL** option. 2. Select the metadata from the **Metadata URL** list. **Tip** If the metadata you want is not shown in the list, click **Manage Partner Metadata URLs**. For more information, see Manage Partner metadata URLs. 3. Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. **Note** A warning will display if you do not have runtime notifications enabled. To enable these notifications, go to **System > Monitoring & Notifications > Runtime Notifications** and select the **Notification for SAML Metadata Update Events** box. 4. Click **Load Metadata**. **Note** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note** If there is a digital signature error, click **Manage Partner Metadata URLs** to resolve the issue.

2. Click **Next**.

Identifying the partner

When creating an identity provider (IdP) connection, you must identify your partner and provide basic information about them.

About this task

On the **General Info** tab, you provide your partner’s unique federation identifier, the display name of the connection, and some other optional information, such as virtual server IDs, contact information, and logging mode for runtime transaction logging.

In addition, on this tab you can define a default error message that end users see in the event that single sign-on (SSO) fails.

Steps

1. Provide the basic information to identify your partner.

See the following table for more information.

Field	Description
Partner's Entity ID, **Issuer**, **Partner's Realm**, or **Connection ID** (Required)	The published, protocol-dependent, unique identifier of your partner. For a SAML 2.0 connection, this is your partner's SAML Entity ID. For a SAML 1.x connection, this is the audience your partner advertises. This ID might have been obtained out-of-band or through a SAML metadata file. For a WS-Federation connection, this is your partner's Realm. For an OpenID Connect connection, this is the Issuer Identifier of the OpenID Provider (OP). For a security token service (STS)-only connection, this ID can be any unique identifier.
Enable Additional Issuers (Applicable only to OpenID Connect connection)	When selected, PingFederate takes into consideration additional issuers when validating ID tokens obtained through this connection. **Tip** Enable this option if you want to support multi-tenant OpenID providers, such as Microsoft Azure AD. This check box is not selected by default. Issuer information is defined on the **Additional Issuers** tab.
Connection Name (Required)	A plain-language identifier for the connection: for example, a company or department name. This name is displayed in the connection list on the administrative console.
Virtual Server IDs (Not applicable to OpenID Connect connections)	If you want to identify your server to this connection partner using an ID other than the one you specified on the **Federation Info** tab, enter a virtual server ID in this field and click **Add**. Enter additional virtual server IDs as needed.
Client ID and **Client Secret** (Applicable to and required for OpenID Connect connections)	The client ID and the client secret to communicate with the OP. This client represents PingFederate and is created and managed at the OP. For more information, see the documentation provided by the OP.
Base URL	The fully qualified hostname and port on which your partner's federation deployment runs (for example, `https://www.example.com:9031`). This entry is an optional convenience, allowing you to enter relative paths to specific endpoints, instead of full URLs, during the configuration process.
Company	The name of the partner company to which you are connecting.
Contact Name	The contact person at the partner company.
Contact Number	The phone number of the contact person at the partner company.
Contact Email	The email address for the contact person at the partner company.

Field	Description
Error Message (Applicable only to SAML or OpenID Connect connections)	If an error occurs on this server, the end user's browser might be redirected in a default situation to an error page hosted within PingFederate. The default entry `errorDetail.spSsoFailure` is a variable from the `<pf_install>/pingfederate/server/default/conf/language-packs/pingfederate-messages.properties` file, which is part of the PingFederate localization framework. If localization is not needed, you can also specify a message directly in this field to change the default.
Logging Mode	The level of transaction logging applicable for this connection. The default selection is **Standard**.

If the OP supports the OpenID Connect Discovery specification, the connection setup is expedited by loading the metadata from the OP. Based on the discovery specification, PingFederate makes a direct HTTP GET request to the `/.well-known/openid-configuration` endpoint at the OP and populates the attribute contract and the protocol settings of the connection automatically. Manual adjustments can be made during the connection setup or at a later time.

Additionally, you can refresh the protocol settings by reloading the metadata from the OP at any time. If additional claims are supported, PingFederate adds them to the attribute contract, so that they can be mapped to the target applications later. In the event that the previously supported claims have been dropped by the OP from the metadata, they remain in the attribute contract. While the existing mapping configuration might not be adversely affected, runtime errors might occur if certain attributes are no longer available to the target applications. If runtime errors occur, review the server log and modify the mapping configuration accordingly.

2. **Optional:** Click **Load Metadata**.

   This step is not applicable to an STS-only connection.

Populating extended property values for IdP connections

Add, remove, or update extended property values.

Steps

1.
2. On the **System** tab, add, remove, or update one or more values for any extended properties defined in **System > Server > Extended Properties**.

   > **Note**
   >
   > Any values defined for these extended properties will be passed to all applicable velocity templates and as a request context parameter in the authentication API.

   Extended property values can serve as metadata. They can also help drive authentication requirements. For more information, see Extended properties.

Defining additional issuers

You can define additional issuers when creating an identity provider (IdP) connection.

About this task

On the **Additional Issuers** tab, define additional issuers that PingFederate can accept when validating ID tokens obtained through this connection. This tab appears only when the **Enable Additional Issuers** check box on the **General Info** tab is selected.

Steps

1. Go to the **Additional Issuers** tab.
2. **Optional:** Select the **Accept All Issuers (Not Recommended)** check box if you want PingFederate to accept any issuers when validating ID tokens obtained through this connection.

   > **Caution**
   >
   > As suggested by the property name, we do not recommend accepting any issuers.

   This check box is not selected by default.
3. **Optional:** Define additional issuers.

   Applicable only when the **Accept All Issuers (Not Recommended)** check box is not selected.

   1. Enter the issuer under **Additional Issuer**.
   2. **Optional:** Enter information about the issuer under **Description**.

   The **Primary Issuer** field represents the issuer defined on the **General Info** tab and is always accepted.

Configure SP Browser SSO

PingFederate supports multiple configurations for browser single sign-on (SSO) with different federation standards. You can configure these options from the **Browser SSO** tab.

Browser single sign-on (SSO) relies on a user's web browser and HTTP requests to broker identity-federation messaging in XML or JSON web tokens (JWT) between an identity provider (IdP) and a service provider (SP). In contrast, WS-Trust security token service (STS) messaging is typically application-driven across the back channel and does not require browser mediation.

> **Tip**
>
> Many steps involved in setting up a federation connection are protocol-independent; that is, they are required steps for all connections, regardless of the associated standards. For more information, see Federation roles. Also, for any given connection, some configuration steps are required under the applicable protocol, while others are optional. Still others are required only based on certain selections. The administrative console determines the required and optional steps based on the protocol and dynamically presents additional requirements or options based on selections.
> The following sections provide sequential information about every step you might encounter while configuring browser-based SSO, regardless of the protocol you are using for a particular connection.

SAML 2.0 configuration steps

- Selecting SAML profiles
- Configuring user-session creation
    - Choosing an identity mapping method for SP SSO
    - Defining an attribute contract
    - Managing target session mappings
- Configuring protocol settings
    - Specifying SSO service URLs (SAML)
    - Defining SLO service URLs (SAML 2.0)
    - Selecting allowable SAML bindings (SAML)
    - Specifying an artifact lifetime (SAML 2.0)
    - Defining artifact resolver locations (SAML)
    - Configuring default target URLs
    - Overriding authentication context in an IdP connection
    - Configuring signature policy
    - Specifying XML encryption policy (for SAML 2.0)

SAML 1.x configuration steps

- Selecting SAML profiles
- Configuring user-session creation
    - Choosing an identity mapping method for SP SSO
    - Defining an attribute contract
    - Managing target session mappings
- Configuring protocol settings
    - Specifying SSO service URLs (SAML)
    - Selecting allowable SAML bindings (SAML)
    - Defining artifact resolver locations (SAML)
    - Configuring default target URLs
    - Configuring signature policy

WS-Federation configuration steps

- Configuring user-session creation
  - Choosing an identity mapping method for SP SSO
  - Defining an attribute contract
  - Managing target session mappings
- Configuring protocol settings
  - Specifying a service URL (WS-Federation)
  - Configuring default target URLs
  - Configuring signature policy

OpenID Connect configuration steps

- Configuring user-session creation
  - Choosing an identity mapping method for SP SSO
  - Defining an attribute contract
  - Managing target session mappings
- Configuring protocol settings
  - Configuring OpenID Provider information
  - Configuring default target URLs
  - Overriding authentication context in an IdP connection

After configuring SSO settings, you will need to configure authentication credentials, the range of which depends on your SSO selections. For more information, see Configuring security credentials. Also, other configuration tasks might remain to be configured for new or modified connections, depending on the selected options on the **Connection Options** tab.

Selecting SAML profiles

A SAML profile is the message-interchange scenario that you and your federation partner agree to use. It defines the settings that support SAML usage for applications.

About this task

For SAML 2.0, PingFederate supports all identity provider (IdP) and service provider (SP)-initiated single sign-on (SSO) and single logout (SLO) profiles. For SAML 1.x, PingFederate supports both the standard IdP-initiated SSO profile and a proprietary "destination-first" SP-initiated SSO profile.

> **Note**
>
> When configuring a local loopback connection, in which one PingFederate instance is both the identity provider and the service provider, disable the IdP-Initiated SLO and SP-Initiated SLO options on the Browser SSO window's SAML Profiles tab. These options determine whether SAML logout requests should be sent to the partner during the SLO flow. Those requests aren't necessary and can cause unexpected behavior when the partner connection exists locally. All local sessions for loopback connections are terminated during the SLO flow without the need to send SAML requests.

For information on typical SAML SSO and SAML 2.0 SLO profile configurations, including illustrations, see SAML 1.x profiles and SAML 2.0 profiles.

> **Note**
>
> The **SAML Profiles** tab does not apply to OpenID Connect and WS-Federation IdP connections.

Steps

- Select the applicable profiles based on your partner agreement.

  For SAML 2.0, you must select at least one SSO profile.

  For SAML 1.x, IdP-initiated SSO is assumed and the specifications do not support SLO; the only choice on this tab is **SP-initiated SSO**.

Configuring user-session creation

You must create and configure user sessions when configuring service provider (SP) browser single sign-on (SSO).

About this task

As an SP, you must specify how PingFederate uses information sent from the IdP in SSO tokens to create user sessions for enabling access to protected resources at your site.

If you are a federation hub, bridging an identity provider to one or more service providers, you can associate one or more authentication policy contracts to the IdP connection. For more information, see Federation hub use cases.

The configuration involves choosing an identity-mapping method, establishing an attribute contract as needed, and optionally mapping one or more SP adapter instances, authentication policy contracts, or both.

Steps

- On the **User-Session Creation** tab, click **Configure User-Session Creation**.

Choosing an identity mapping method for SP SSO

When configuring service provider (SP) single sign-on (SSO), PingFederate offers two methods of identity mapping you can choose from: account mapping or account linking.

About this task

PingFederate allows an SP to use either account linking or account mapping to associate remote users with local accounts for SSO between business partners. For more information, see Identity mapping. On the **Identity Mapping** tab, you choose which method to use in this IdP connection. You and your partner should decide in advance which option to use. For more information, see Federation planning checklist.

If your site is using account linking, then establishing an attribute contract is not required. Depending on your partner agreement, you can choose to supplement the account link with an attribute contract. In this configuration the account link is used to determine the user's identity, while the additional attributes might be used for authorization decisions, customized web pages, and so on, at the your site. For more information, see User attributes.

> **Important**
>
> If you have previously set up a configuration to use an attribute contract and want to change the configuration to use account linking without additional attributes, then the existing attribute contract will be discarded.

Account linking can be used with either a clear, standard name identifier or an opaque pseudonym.

Steps

1. Choose which identity mapping method to use in this IdP connection.

   Choose from:

   - If you want to dynamically associate remote users with local accounts using a known attribute to identify a user, such as a username or email address, select **Account Mapping**

     Account mapping uses the user identifier, `SAML_SUBJECT` in a SAML assertion or `sub` in an ID token, and associated user attributes to create an association between a remote user and a local account.

     > **Tip**
     >
     > If you are using PingFederate's JIT provisioning, choose **Account Mapping**. For more information, see Configuring just-in-time provisioning.

   - If you want to create a long-term association between a remote user and a local account, select **Account Linking**

     > **Caution**
     >
     > Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
     > Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

   To set up an attribute contract to use in conjunction with account linking, select the **... includes attributes in addition to the unique name identifier** check box.

2. If you have selected only the SP-initiated SSO profile and you intend to enforce additional authentication requirements by placing this IdP connection in an SP authentication policy, select **No Mapping**.

3. Additionally, select **No Mapping** if you are deploying an IdP connection solely for OAuth attribute mapping without the use of an authentication policy contract. For more information, see Configuring IdP connection grant mapping.

Defining an attribute contract

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in single sign-on (SSO) tokens for this connection.

About this task

You can extend the attribute contract with additional attributes. Optionally, you can configure PingFederate to mask individual extended attributes in its logs. For more information, see Attribute contracts and Attribute masking.

> **Tip**
>
> If you are creating or updating a SAML or an OpenID Connect identity provider (IdP) connection, consider using the partner's metadata to do so. If the metadata contains the required information, PingFederate automatically populates the attribute contract for you.

Steps

1. On the **Attribute Contract** tab, enter the attribute name in the text box.

   Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.

Tip

If you are configuring a SAML connection to an InCommon participant, the assertion might contain attributes such as `urn:oid:0.9.2342.19200300.100.1.3` and `urn:oid:2.5.4.42`, which are standard names under various specifications, such as RFC4524 andhttps://tools.ietf.org/html/rfc4519[RFC4519]. For more information, see www.incommon.org/participants. The following table describes a subset of the object IDs (OIDs) referenced by the most common attributes used by InCommon participants.

OID value	Description
0.9.2342.19200300.100.1.3	`mail`
1.3.6.1.4.1.5923.1.1.1.1	`eduPersonAffiliation`
1.3.6.1.4.1.5923.1.1.1.6	`eduPersonPrincipalName`
1.3.6.1.4.1.5923.1.1.1.7	`eduPersonEntitlement`
1.3.6.1.4.1.5923.1.1.1.9	`eduPersonScopedAffiliation`
1.3.6.1.4.1.5923.1.1.1.10	`eduPersonTargetedID`
2.5.4.3	`cn`
2.5.4.4	`sn`
2.5.4.10	`o`
2.5.4.42	`givenName`
2.16.840.1.113730.3.1.241	`displayName`

For other attributes, see the metadata from your partner. The `FriendlyName` values, if available, should provide additional information about the attributes. Alternatively, third-party resources, such as www.ldap.com/ldap-oid-reference and www.oid-info.com, might help as well.

2. **Optional:** Select the check box under **Mask Values in Log**.

3. Click **Add**.

4. Repeat until all desired attributes are defined.

Next steps

Click **Edit**, **Update**, and **Cancel** to make or undo a change to an item. Click **Delete** and **Undelete** to remove an item or cancel the removal request.

Managing target session mappings

You can map a service provider (SP) adapter instance to an identity provider (IdP) connection and complete its mapping configuration through a series of sub tasks.

About this task

When PingFederate receives an SSO token, the corresponding SP adapter is triggered to fulfill its adapter contract based on the connection settings for the purpose of completing the "last-mile" integration with your application. As needed, you can map multiple SP adapter instances to an IdP connection, the same SP adapter instance to multiple IdP connections, or a combination of them.

Alternatively, if you use authentication policies to route users through a series of authentication sources and end each successful policy path with an authentication policy contract (APC), you can skip the mapping of an APC to an IdP connection and configure an APC-to-SP adapter instance mapping configuration.

> **Tip**
>
> To learn more about authentication policies, see Authentication policies.

Furthermore, you can map one or more APCs to an IdP connection to bridge an identity provider to one or more service providers. In this scenario, PingFederate is a federation hub for both sides. PingFederate uses APCs to associate this IdP connection with the applicable SP connections to the service providers; each APC has its own set of attributes to which you can map values from the SSO tokens.

> **Tip**
>
> To learn more about federation hub, see Federation hub use cases.

On the **Target Session Mapping** tab, if presented, you must associate at least one target session, an SP adapter instance or an authentication policy contract, with an IdP connection. If you have multiple integration requirements, for example, if you are using more than one IdM system or an application not covered by a centralized system, multiple SP adapter instances. If you are bridging an identity provider to multiple service providers, map multiple authentication policy contracts.

The **Target Session Mapping** configuration does not apply when the **No Mapping** option is selected on the **Identity Mapping** tab.

Steps

- On the **Target Session Mapping** tab:

Choice	Action
Map an SP adapter instance	Click **Map New Adapter Instance**.
Map an APC	Click **Map New Authentication Policy**.
Edit the mapping configuration of an SP adapter instance or APC	Open it by clicking on its name, select the setting that you want to reconfigure, and complete the change.

Choice	Action
Remove an SP adapter or APC or cancel the removal request	Click **Delete** followed by **Save** or **Undelete**.
If you are creating a new connection and you are finished with mapping the required target sessions	Click **Done**.
If you are editing an existing configuration and want to keep your changes	Click **Save**.

Result

When target sessions are restricted to certain virtual server IDs, the allowed IDs are displayed under **Virtual Server IDs**.

> **Note**
>
> If you configure multiple target sessions for a connection, PingFederate selects the applicable adapter instance or authentication policy contract at runtime based on the target resource information in the requests and your configuration For more information, see Configuring target URL mapping.

Selecting a target session

The first step of the mapping configuration is to map an adapter instance or an authentication policy contract to your connection.

Steps

1. Select an adapter instance from the **Adpater Instance** list.

   If you do not see the desired adapter instance, click **Manage Adapter Instances** to create a new instance of any deployed adapter.

2. If you want to customize adapter settings for this connection alone, select the **Override Instance Settings** check box.

   Result:

   When selected, the administrative console adds a new set of sub tasks: the **Override Instance** tab and its sub tasks.

   > **Tip**
   >
   > Alternatively, you can create child adapter instances of a base adapter instance with overrides so that such customized settings can be applied to several connections. For more information, see Hierarchical plugin configurations.

3. **Optional:** To map an authentication policy contract (APC), select an adapter instance from the list.

4. **Optional:** If you do not see the desired APC, click **Manage Authentication Policy Contracts** to create a new policy contract.

Result

If you are editing a currently mapped adapter instance, you can select the **Override Instance Settings** check box. Clearing it removes all previously overridden settings for this connection. Selecting it provides you the opportunity to customize adapter settings specifically for this connection.

If you are editing a currently mapped APC, no changes can be made on this tab.

Overriding an SP adapter instance

When configuring service provider (SP) browser single sign-on (SSO), you can override the adapter instance settings.

About this task

On the **Override Instance** tab, you start a series of sub tasks to override adapter settings specifically for this connection.

> **Note**
>
> Any changes to the base adapter instance are propagated to a connection provided the same changes are not overridden for the connection.

Steps

- Click **Override Instance Settings**.

  On each of the setting tabs, select the **Override** check box, make your changes, then click **Next**. When you are finished, click **Done** to continue with the rest of the mapping configuration.

  > **Note**
  >
  > The override setting tabs are functionally identical to those used for creating a new adapter instance. For more information, see Managing SP adapters.

Next steps

If you are editing a currently mapped adapter instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You can also remove all overridden settings on a per-tab basis by clearing the **Override** check box near the top of the tab.

Restricting a target session to certain virtual server IDs

You can enforce integration requirements by restricting a target session to certain virtual server IDs when you multiplex one connection for multiple environments.

About this task

For more information, see Multiple virtual server IDs. On the **Virtual Server IDs** tab, follow these steps to restrict a target session to a certain virtual server ID. By default, no restriction is imposed.

Steps

1. Select the **Restrict Virtual Server IDs** check box.
2. Select thevirtual server IDs that you want to allow for this target session.

Result

If you are editing a currently mapped adapter instance or authentication policy contract (APC), you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Choosing an attribute mapping method

You can select if and how PingFederate should query a local datastore to help fulfill the attribute contract in conjunction with attribute values from the single sign-on (SSO) token.

Before you begin

To determine whether you need to look up additional values, compare the attribute contract against the adapter contract or the authentication policy contract. If the attribute contract does not contain the required information, determine whether a local datastore can supply it.

Alternatively, you can configure datastore queries as part of the fulfillment configuration for the applicable APC if you use authentication policies to route users through a series of authentication sources and end each successful policy path with an APC.

About this task

You make selections on the **Adapter Data Store** tab for service provider (SP) adapter mapping or the **Attribute Retrieval** tab for authentication policy contract (APC) mapping.

> **Tip**
>
> To learn more about authentication policies, see Authentication policies.

Steps

- If the attribute contract contains all the attributes that your application requires, click **Use only the attributes available in the SSO assertion**.
- To set up a datastore query, click **Use the SSO assertion to look up additional information**, and then follow a series of sub tasks to complete the configuration. See Choosing a datastore for step-by-step instructions.

Result

If you are editing a currently mapped adapter instance or APC, you can change the mapping method, which might require additional configuration changes in subsequent tasks.

Configuring target session fulfillment

Map values to the attributes defined for the contract. These are the values that the target application requires to create a local session for the user.

Before you begin

If you are bridging an identity provider (IdP) to one or more service providers, the values mapped to the authentication policy contracts are used by the associated service provider (SP) connections to create assertions for the service providers. For more information, see Federation hub use cases.

At runtime, a single sign-on (SSO) operation fails if PingFederate cannot fulfill the required attribute.

Steps

1. On the **Adapter Contract Fulfillment** tab, for each attribute, select a source from the **Source** list and then choose or enter a value. You must map all attributes.
   - **AccountLink**

When selected, the **Value** list populates with **Local User ID**. Normally, you would map **Local User ID** to an adapter attribute that represents the user identifier at the target. This source is not applicable to authentication policy contracts. This source appears only if you have elected to use account linking for a target session on the **Identity Mapping** tab.

- **Assertion** or **Provider Claims**

  When selected, the **Value** list populates with attributes from the SSO token. Select the desired attribute from the list.

  For example, to map the value of `SAML_SUBJECT` from a SAML assertion as the value of the `subject` user identifier on the contract, select **Assertion** from the **Source** list and **SAML_SUBJECT** from the **Value** list.

Context

When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

> **Note**
>
> As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

> **Note**
>
> If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
>
> - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
> - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
>   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
>   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
>   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
> - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field.
>   This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

+ **[.uicontrol]**LDAP**, [.uicontrol]**JDBC**, or [.uicontrol]**Other**

+ When selected, the **Value** list populates with attributes that you have selected from the datastore. Select the desired attribute from the list.

+

- **Expression**

  When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select **Expression** from the **Source** list, click **Edit** under **Actions**, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see **Text**.

Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- **No Mapping**

  Select this option to ignore the **Value** field.

- **Text**

  When selected, the text you enter is used at runtime. You can mix text with references to any of the values from the SSO token, using the `$\{attribute}` syntax.

  You can also enter values from your datastore, when applicable, using this syntax:

  `[.codeph]``$\{ds.[.varname]__attribute__}```

  where `attribute` is any attribute that you have selected from the datastore.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.
> Two other text variables are available. `SAML_SUBJECT` is the initiating user or other entity. `TargetResource` is a reference to the protected application or other resource for which the user requested SSO access. The `${TargetResource}` text variable is available only if specified as a query parameter for the relevant endpoint, either as `TargetResource` for SAML 2.0 or `TARGET` for SAML 1.x.

You might hard-code a text value for a variety of reasons.For example, if your web application provides a consumer service, you might want to supply a particular promotion code for the partner.

> **Note**
>
> If you are editing a currently mapped adapter instance or authentication policy contract (APC), you can update the mapping configuration, which might require additional configuration changes in subsequent tasks.

1. Click **Next** to continue configuration.

Defining issuance criteria for SP Browser SSO

You can define issuance criteria when configuring service provider (SP) browser single sign-on (SSO) for PingFederate.

Before you begin

Begin this optional configuration by adding a criterion. Choose the source that contains the attribute to be verified. Some sources, such as **Mapped Attributes**, are common to almost all use cases. Other sources, such as **JDBC**, depend on the type of configuration. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to be verified. Depending on the selected source, the available attributes or properties vary. Finally, specify the comparison method and the desired, compared-to, value.

About this task

On the **Issuance Criteria** tab, define the criteria that must be satisfied in order for PingFederate to process a request further. This token authorization feature provides the capability to conditionally approve or reject requests based on individual attributes.

If you define multiple criteria, all criteria must be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value depending on the chosen comparison method. The multi-value contains and multi-value does not contain comparison methods are intended for attributes that might contain multiple values. Such criterion is considered satisfied if one of the multiple values matches or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Important**
>
> When you multiplex one connection for multiple environments, consider using attribute mapping expressions to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access. For more information, see Multiple virtual server IDs and Issuance criteria and multiple virtual server IDs.

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the **Source** list, select the attribute's source.

Depending on the selection, the **Attribute Name** list populates with associated attributes. See the following table for more information.

Source	Description
AccountLink	Select to evaluate the **Local User ID** value of the user. Displayed only if **Account Linking** is the selected identity mapping method. For more information, see Choosing an identity mapping method for SP SSO.
Assertion or **Provider Claims**	Select to evaluate attributes from the IdP connection.
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, attribute mapping expressions are more appropriate to evaluate and return values.
JDBC, **LDAP**, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

1. In the **Attribute Name** list, select the attribute to be evaluated.

Available methods:

- **equal to**
- **equal to (case insensitive)**
- **equal to DN**
- **not equal to**
- **not equal to (case insensitive)**
- **not equal to DN**
- **multi-value contains**
- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+ . In the **Value** field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

+ Error results are handled in one of the two ways:

+ Redirect:::

When an `InErrorResource` URL is provided, the value of the **Error Result** field is used by the query parameter `ErrorDetail` in the redirect URL.

+ Template:::

When an `InErrorResource` URL is not provided, the value of the **Error Result** field is used by the variable *$errorDetail* in the `sp.sso.error.page.template.html` template file.

+ Using an error code in the **Error Result** field allows the error template or an application to process the code in a variety of ways, for example, display an error message or e-mail an administrator.

+ To use localized descriptions, enter a unique alias in the **Error Result** field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

+ If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click **Add**.

+ . **Optional:** Repeat to add more criteria.

+ . If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

+ For more information, see Attribute mapping expressions.

1. Click **Show Advanced Criteria**.
2. In the **Expression** field, enter the required expressions.
3. In the **Error Result** field, enter an error code or message.

> **Note**
>
> If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click **Add**.
5. **Optional**: Click **Test**, enter values in the applicable fields, and verify the results.
6. **Optional**: Repeat to add multiple criteria using attribute mapping expressions.

Related links
Reviewing the target session mapping

You can review your target session mapping configuration and settings before completing.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Reviewing the session creation summary

You can review your session creation settings before completing configuration.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configuring protocol settings

The **Protocol Settings** tab on the **Browser SSO** window provides the launching point for configuring partner endpoints, message customizations, and other protocol-specific settings for browser-based single sign-on (SSO) connections.

About this task

The settings available on the **Protocol Settings** tab depend on the connection's protocol. The availability of the settings also depends on the connection's other previously configured settings.

SAML 2.0

- Outbound SSO bindings (POST, redirect, artifact) and the corresponding SSO service URLs
- Outbound single logout (SLO) bindings (POST, redirect, artifact, SOAP) and the corresponding protocol endpoints
- Inbound bindings (POST, redirect, artifact, SOAP)
- Artifact lifetime
- Artifact resolution location
- Default target URL
- Authentication context mappings
- Signature policy
- Encryption policy

SAML 1.x

- Outbound SSO service URL, also known as the Intersite Transfer Service, if the service provider (SP)-Initiated SSO profile is enabled
- Inbound bindings (POST, artifact)

- Artifact resolution location
- Default target URL
- Signature policy

WS-Federation

- Protocol endpoint
- Default target URL
- Signature policy

OpenID Connect

- The scopes PingFederate sends to the OpenID provider (OP) in its authorization and token requests
- The OpenID Connect login type and authentication scheme used by PingFederate when communicating with the OP
- The authorization endpoint, the token endpoint, the user information, UserInfo, endpoint, and the JWKS URL
- Default target URL
- Authentication context mappings

To start configuring the connection's protocol setting:

Steps

1. On the **Protocol Settings** tab, click **Configure Protocol Settings**.

   Result:

   The **Protocol Settings** window opens.

2. Use the window's tabs to continue configuring the connection's protocol settings.

Specifying SSO service URLs (SAML)

The single sign-on (SSO) service endpoint is where PingFederate sends requests when SSO is initiated at your site according to partner requirements. It applies to all SAML versions when the service provider (SP)-initiated SSO profile is enabled.

About this task

For SAML 2.0 connections, associate bindings to the endpoints where your identity provider (IdP) partner wants PingFederate to send authentication requests when SSO is initiated at your site.

For SAML 1.x, only one endpoint is allowed, and the binding selection is not required.

Some federation use cases might require additional customizations in the authentication requests sent from the PingFederate SP server to the IdP, such as including the optional `Extensions` element in the authentication requests. You can use OGNL expressions to fulfill these use cases.

Steps

1. Enter an SSO service endpoint.
    1. Enter the SSO service endpoint in the **Endpoint URL** field.

       You can enter a relative path, starting with a forward slash, if you have provided a base URL on the **General Info** tab.

       For SAML 1.x connections, this is the only configurable item on the **SSO Service URL** tab.

       The remaining steps on the **SSO Service URLs** tab only apply to SAML 2.0 connections.
    2. Select a SAML binding from the list; for example, **POST**.
    3. Click **Add**.
    4. **Optional:** Repeat to add additional SSO service endpoints.
2. **Optional:** Customize messages using OGNL expressions.

   Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions. Additionally, message customization does not apply to SAML 1.x connection.
    1. Click **Show Advanced Customizations**.
    2. Select a message type from the list.
    3. Enter an OGNL expression to fulfill your use case.

       > **Note**
       >
       > For more information about **Message Type**, available variables, and sample OGNL expressions, see Customizing assertions and authentication requests.
    4. Click **Add**.
    5. **Optional:** Repeat to add another message customization.

Specifying a service URL (WS-Federation)

The service endpoint URL is where PingFederate sends request for security token (RST) and single log-out (SLO) messages.

About this task

To protect against session token hijacking, PingFederate provides an option to validate wreply for SLO. When this option is enabled, you can specify additional allowed domains and paths on this tab. PingFederate validates the locations against a consolidated list of allowed domains and paths from all active WS-Federation connections before redirecting the end users to their destinations.

> **Note**
>
> The settings to enter additional allowed domains and paths appear only if the option to validate wreply for SLO is enabled. For more information, see Managing partner redirect validation.

Steps

1. Enter the WS-Federation protocol endpoint at the identity provider (IdP) site in the **Endpoint URL** field.

   You can enter a relative path, starting with a forward slash, if you have provided a base URL on the **General Info** tab.

2. **Optional:** Specify additional allowed domains and paths.

   1. Indicate whether to mandate secure connections when this resource is requested under **Require HTTPS**.

      > **Important**
      >
      > This selection is recommended to ensure that the validation will always prevent message interception for this type of potential attack, under all conceivable permutations.

      This check box is selected by default.

   2. Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

      Enter a value without the protocol, such as `example.com` or `10.10.10.10`.

      Prefix a domain name with a wildcard followed by a period to include subdomains using one entry. For instance, `*.example.com` covers `hr.example.com` or `email.example.com` but not `example.com`, the parent domain.

      > **Important**
      >
      > While using an initial wildcard provides the convenience of allowing multiple subdomains using one entry, consider adding individual subdomains to limit the redirection to a list of known hosts.

   3. **Optional:** Enter the exact path of this resource under **Valid Path**.

      Start with a forward slash, without any wildcard characters in the path. If left blank, any path under the specified domain or IP address is allowed. This value is case-sensitive. For instance, `/inbound/Consumer.jsp` allows `/inbound/Consumer.jsp` but rejects `/inbound/consumer.jsp`.

      You can allow specific query parameters with or without a fragment by appending them to the path. For instance, `/inbound/Consumer.jsp?area=West&team=IT#ref1001` matches `/inbound/Consumer.jsp?area=West&team=IT#ref1001` but not `/inbound/Consumer.jsp?area=East&team=IT#ref1001`.

   4. Select the check box under **Allow Any Query/Fragment** to allow any query parameters or fragment for this resource.

      Selecting this check box also means that no query parameter and fragment are allowed in the path defined under **Valid Path**.

This check box is not selected by default.

Defining SLO service URLs (SAML 2.0)

On the **SLO Service URLs** tab, associate bindings to the endpoints where your identity provider (IdP) receives logout requests when single logout (SLO) is initiated at your site and where PingFederate sends SLO responses when it receives SLO requests from the IdP.

About this task

This process only applies to SAML 2.0 connections when either SLO profile is selected on the **SAML Profiles** tab.

Steps

1. Go to **Applications** > **Integrations** > **SP Connections.**
2. Click on any SAML 2.0 connection, and then click the **Browser SSO** tab.
3. Click **Configure Browser SSO**, and then click the **SAML Profiles** tab.
4. Select a SAML binding from the list; for example, **POST**.
5. Enter the SLO endpoint URL in the **Endpoint URL** field.

   You can enter a relative path, starting with a forward slash, if you have provided a base URL on the **General Info** tab.
6. **Optional:** Enter a URL in the **Response URL** field.

   When specified, it is the location where SLO logout response messages are sent based on your partner agreement. When omitted, PingFederate sends logout responses to the SLO endpoint URL.

   You can enter a relative path, starting with a forward slash, if you have provided a base URL on the **General Info** tab.
7. Click **Add**.
8. **Optional:** Repeat to add additional SLO endpoints.

Result

If you are editing an existing connection, you can reconfigure the SLO endpoints, which might require additional configuration changes in subsequent tasks.

Selecting allowable SAML bindings (SAML)

On the **Allowable SAML Bindings** tab, select the one or more bindings that your identity provider (IdP) partner can use to send SAML assertions or SAML 2.0 single logout (SLO) messages.

About this task

This configuration applies to all SAML connections.

Steps

1. Select the check boxes for only the applicable SAML bindings based on your partner agreement.

   Choose from:

   - **Artifact**
   - **Post**
   - **Redirect**
   - **SOAP**

   > **Note**
   >
   > If you have specified a single sign-on (SSO) or SLO endpoint using the artifact outbound binding, you must include SOAP as one of the allowable inbound binding.

Result

If you are editing an existing connection, you can reconfigure the allowable bindings, which might require additional configuration changes in subsequent tasks.

Specifying an artifact lifetime (SAML 2.0)

When PingFederate sends an artifact to your identity provider’s (IdP’s) single sign-on (SSO) or single logout (SLO) service endpoint, an element in the message indicates how long it should be considered valid.

About this task

On the **Artifact Lifetime** tab, specify the expiry information in seconds.

You can change the default value to meet your requirements. You should also consider synchronizing your serve clock with your partner’s SAML gateway server. If clocks are not synchronized, you might need to set the artifact lifetime to a higher value to prevent latency issues.

This step applies only to SAML 2.0 connections.

Steps

- **Optional:** Override the default value of the **Artifact Lifetime** field.

  The default value is 60 seconds.

Result

You can update the artifact lifetime if you are editing an existing connection.

Defining artifact resolver locations (SAML)

When you enable the artifact binding as one of the allowable bindings on the **Allowable SAML Bindings** tab, you must provide an artifact resolution service (ARS) endpoint.

About this task

The ARS endpoint is the location where PingFederate sends back-channel requests to resolve artifacts received from the identity provider (IdP).

SAML 2.0 connections allow multiple ARS endpoints. For SAML 1.x connections, you can only enter one ARS endpoint.

Steps

1. Enter an ARS endpoint.

    1. Enter the ARS endpoint URL.

        You can enter a relative path, starting with a forward slash, if you provide a base URL on the **General Info** tab.

        Result:

        If you are configuring a SAML 1.x connection, you can only enter one ARS endpoint on the **Artifact Resolver Location** tab.

    2. **Optional:** Enter an integer in the **Index** field for this ARS endpoint.

        This is applicable only to SAML 2.0 connections.

        The administrative console automatically assigns an index value for each ARS endpoint, starting from 0. If you want to define your own index values, you must make sure the index values are unique.

    3. Click **Add**.

    4. **Optional:** Repeat to add additional ARS endpoints.

        This is applicable only to SAML 2.0 connections.

        > **Note**
        >
        > When specifying multiple ARS endpoints, each endpoint must share the same transport protocol. That is, if one endpoint uses HTTPS, then all must use HTTPS. Similarly, if one endpoint uses HTTP, then all must use HTTP.

2. **Optional:** Enter your partner's source ID.

    The source ID is usually a generated value based on a federation partner's connection ID; the PingFederate service provider (SP) server will correctly generate the source ID. If that is the case for this partner, then leave this field blank. If your partner uses a Source ID that is not based on the Issuer ID, then enter the Source ID supplied by your IdP partner.

Result

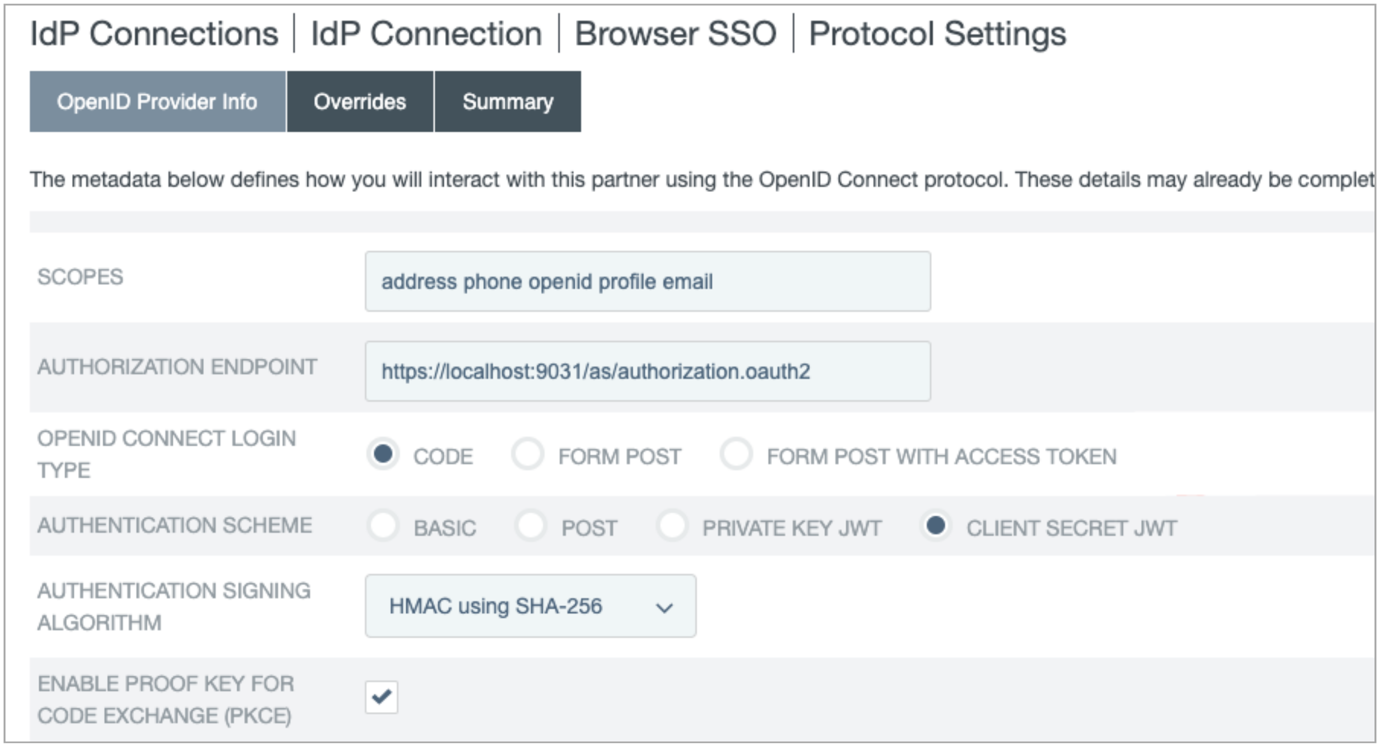
You can reconfigure any ARS endpoint or the source ID value for SAML 1.x if you are editing an existing connection.

Configuring OpenID Provider information

You must configure OpenID Provider (OP) settings and information when configuring service provider (SP) browser single sign-on (SSO).

Steps

1. On the **OpenID Provider Info** tab, provide the scopes, the endpoints, and the authentication scheme.



IdP Connections | IdP Connection | Browser SSO | Protocol Settings

The metadata below defines how you will interact with this partner using the OpenID Connect protocol. These details may already be complet

SCOPES: address phone openid profile email

AUTHORIZATION ENDPOINT: https://localhost:9031/as/authorization.oauth2

OPENID CONNECT LOGIN TYPE: ● CODE ○ FORM POST ○ FORM POST WITH ACCESS TOKEN

AUTHENTICATION SCHEME: ○ BASIC ○ POST ○ PRIVATE KEY JWT ● CLIENT SECRET JWT

AUTHENTICATION SIGNING ALGORITHM: HMAC using SHA-256

ENABLE PROOF KEY FOR CODE EXCHANGE (PKCE): ☑

> **Note**
>
> If you clicked **Load Metadata** from the OpenID Provider (OP) on the **General Info** tab, the **Scopes** field and all endpoints are pre-populated, provided that the metadata contains the information.

Field	Description
Scopes	The scopes to be included in the OpenID Connect (OIDC) authentication and OAuth token requests to the OP. Multiple space-separated values are allowed. The default value, without loading metadata from the OP, is `openid`. **Tip** You can find a list of OIDC defined scopes in Requesting Claims using Scope Values in the OIDC specification.

Field	Description
Authorization Endpoint	The authorization endpoint at the OP. You can enter a relative path, starting with a forward slash, if you provide base URL on the **General Info** tab. There is no default value without loading metadata from the OP.
OpenID Connect Login Type	The OIDC client profile of the client. This client represents PingFederate and is created and managed at the OP. ◦ If the client is configured to support the Basic Client profile, select **Code**. The resulting value of the `response_type` parameter is `code`. * If the client is configured to support the Implicit Client profile, select **Form POST**. The resulting value of the `response_type` parameter is `id_token`. * If the client is configured to support the Implicit Client profile and the target application requires the associated access token, select **Form POST with access token**. The resulting values of the `response_type` parameter are `id_token token`. The default selection, without loading metadata from the OP, is **Code**.
Authentication Scheme	The client authentication method that PingFederate uses. Applicable and visible only to clients supporting the Basic Client profile. ◦ Select **Basic** to submit credentials with HTTP Basic authentication. ◦ Select **POST** to submit credentials with POST. ◦ Select **Private Key JWT** to authenticate with the `private_key_jwt` Client Authentication method. Learn more in Client Authentication in the OIDC specification. ◦ Select **Client Secret JWT** to authenticate with the `client_secret_jwt` Client Authentication method. Learn more in Client Authentication in the OIDC specification. The default selection, without loading metadata from the OP, is **Basic**.

Field	Description
Authentication Signing Algorithm	If **Private Key JWT** or **Client Secret JWT** is the chosen authentication scheme, select the algorithm that PingFederate uses to sign the JSON Web Token (JWT). If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect (OIDC), additional RSASSA-PSS signing algorithms become available for selection. You can find more information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect. ⓘ **Note** If static keys for OAuth and OIDC are enabled, Elliptic-curve cryptography (EC) algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration might affect runtime transactions and require additional changes here. Learn more in Keys for OAuth and OpenID Connect. ⓘ **Note** Based on the chosen signing algorithm, PingFederate selects the signing JSON Web Key (JWK) from its JWK Set (JWKS) at runtime. For the OP to validate the signed JWT, ensure that the OP can access the PingFederate JWKS endpoint, which returns the current JWKS. The PingFederate JWKS endpoint is located at `<Base URL>/pf/JWKS`, where **Base URL** is defined on **System > Server > Protocol Settings > Federation Info**. For example, if the **Base URL** field value is https://www.example.com, the PingFederate JWKS endpoint is https://www.example.com/pf/JWKS. You can pass the PingFederate JWKS endpoint directly to the OP or have the OP contact the PingFederate OP configuration endpoint to obtain the information. Lear more in OpenID Provider configuration endpoint. If **Client Secret JWT** is the chosen authentication scheme, the signing algorithms are `HS256`, `HS384`, and `HS512`.
Enable Proof Key for Code Exchange (PKCE)	Select this checkbox to enable PingFederate to send a SHA256 code challenge and corresponding code verifier as a Proof Key for Code Exchange (PKCE) to the OP during the Code authentication flow. This check ox is applicable and visible only when the **OpenID Connect Login Type** is **Code**. ⓘ **Note** When **Load Metadata** on the **General Info** tab is clicked, PingFederate displays the **Enable PKCE** checkbox if S256 is listed as a supported method in the `code_challenge_methods_supported` by the OP.

Field	Description
Pushed Authorization Request Endpoint	The Pushed Authorization Request (PAR) endpoint at the OP. When you configure a PAR endpoint, the IdP connection sends authorization requests directly to this endpoint. All parameters associated with an authorization request are transmitted to the PAR endpoint. You can find more information about the PAR protocol in OAuth 2.0 Pushed Authorization Requests on the IETF website. You can enter the relative path, `/par`, starting with a forward slash if you provide the base URL on the General Info tab. **Note** If you clicked **Load Metadata** from the OP on the **General Info** tab, the **Pushed Authorization Request Endpoint** field is pre-populated, provided that the metadata contains the information. As such, PAR requests are the new default behavior.
Token Endpoint, **UserInfo Endpoint**, and **JWKS URL**	OAuth 2.0 and OIDC 1.0 endpoints at the OP. Learn more at openid.net/connect. ***Token Endpoint*** The **Token Endpoint** field is only visible and required for clients supporting the Basic Client profile. In other words, the **OpenID Connect Login Type** field is set to **Code**. ***UserInfo Endpoint*** The **UserInfo Endpoint** field is optional. If omitted, PingFederate only has access to the end-user claims from the ID tokens. ***JWKS URL*** The **JWKS URL** is required for PingFederate to validate the inbound ID tokens from the OP. If the OP signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment, or integrated with a hardware security module (HSM) and a static-key configuration for OAuth and OIDC. You can find more information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. There are no default values without loading metadata from the OP.
Sign Request	Select this checkbox to send request parameters as claims in a request object, a self-contained, signed JWT as one `request` query parameter to the OP. When this optional configuration is enabled, the OP can validate the integrity of the request parameters based on the digital signature found in the signed JWT. Learn more in Passing a Request Object by Value in the OIDC specification. When this optional configuration is enabled, the JWT signed request object includes the `jti` (JWT ID) value. This check box is not selected by default, in which case PingFederate sends request parameters with multiple query parameters, unsigned.

Field	Description
Request Signing Algorithm	Select the algorithm that PingFederate uses to sign the request object. Applicable and visible only when the **Sign Request** checkbox is selected. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. You can find more information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect. ⓘ **Note** If static keys for OAuth and OpenID Connect are enabled, Elliptic-curve cryptography (EC) algorithms that have not been configured with active static keys are hidden. Changes made in the static-key configuration might affect runtime transactions and require additional changes here. Learn more in Keys for OAuth and OpenID Connect. ⓘ **Note** PingFederate automatically selects the signing JSON web key (JWK) based on the selected signing algorithm from its JWK Set (JWKS). In order for the OP to validate the signed request object, ensure that the OP can access your PingFederate's JWKS URL, which returns the current set of JSON web keys. The PingFederate JWKS URL is located at `<Base URL>/pf/JWKS`, where **Base URL** is defined on **System > Server > Protocol Settings > Federation Info**. For example, if the **Base URL** field value is https://www.example.com, the PingFederate JWKS URL is https://www.example.com/pf/JWKS. You can pass the JWKS URL directly to the OP or have the OP contact the PingFederate OpenID Provider configuration endpoint for it. Learn more in OpenID Provider configuration endpoint.
Track User Sessions for Logout	When selected, PingFederate tracks logout entries in the user session so that PingFederate can handle back-channel logout requests. Also, when selected, the **IdP Connection** window's **Summary & Activation** tab displays the connection's **Back-Channel Logout URI**. The checkbox is cleared by default.

2. **Optional:** Remain on the **OpenID Provider Info** tab and specify the request parameters that are allowed to be included in the authentication requests to the OP under **Request Parameters**. Learn more in Configuring request parameters and SSO URLs.

Configuring default target URLs

You can define default target URLs for identity provider (IdP) connections.

About this task

Use the **Protocol Settings > Overrides** tab to assign a default target URL for this IdP connection.

Steps

1. **Optional:** Enter a URL in the **Default Target URL** field.

   If specified, the value overrides the default target setting defined in the **Applications > Integration > SP Default URLs** window.

   > **Note**
   >
   > The SAML 1.x specifications for IdP-initiated single sign-on (SSO) require a target URL to be specified. If the target application is specified in the URL parameter to the `/sp/startSSO.ping` application endpoint, any URL specified in the **Default Target URL** field in this window will not be used for those transactions.

Overriding authentication context in an IdP connection

You can map authentication context values between the local and remote values in an OpenID Connect or a SAML 2.0 identity provider (IdP) connection.

About this task

This optional configuration overrides how authentication context values are communicated with partners in both the authentication or authorization requests and their responses. Any values that are not defined in this configuration are passed through as-is.

As needed, you can use an asterisk, `*`, to match any values, a blank value for a scenario where the partner or the local request does not specify an authentication value, or both.

Steps

1. Go to **Authentication > Integration > IdP Connections**.
2. Click the name of the connection to open it in the **IdP Connection** window.
3. On the **Activation & Summary** tab, scroll down to the **Protocol Settings** section, then click **Overrides**.
4. On the **Overrides** tab, specify the **Local** and **Remote** entry, then click **Add**.
5. Repeat the previous step to define additional mappings.

   Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.
6. Click **Save** to complete the configuration.

   Alternatively, click **Next** to carry on with the rest of the connection settings.

ExampleExample

Suppose you are the service provider (SP) and your target application requires either the `urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos` or `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` authentication context. While the IdP is capable of authenticating its users using a Kerberos-based authentication system, a proprietary identity management system, and a few internal web portals, the authentication context values are different than what your application supports. The authentication context values from the IdP are as follows.

Authentication method	AuthnContextvalues
Kerberos-based authentication system	`KerberosAuth`
Internal web portals	`password`, `portal`, or `web`
Proprietary identity management system	No authentication context information is provided

To override the `AuthnContext` values from the IdP, you can configure the IdP connection with the following authentication context mappings.

Local	Remote
`urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos`	`KerberosAuth`
`urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified`	`*`
`urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified`	

The first entry maps `KerberosAuth` to `urn:oasis:names:tc:SAML:2.0:ac:classes:Kerberos`. The second entry maps any authentication context values including `password` and `portal` to `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified`. The last entry overrides the authentication value to `urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified` in the event that the assertion does not contain any authentication context information.

Configuring signature policy

The **Signature Policy** tab provides options controlling how digital signatures are used for SAML and WS-Federation single sign-on (SSO) messages.

About this task

The choices made on this tab depend on your partner agreement. For more information, see Digital signing policy coordination.

Digital signing is required for SAML response messages sent from the identity provider (IdP) through POST or redirect for SAML 2.0. The SAML specifications allow the signing of the entire SAML response message or the assertion portion inside the SAML response message. If you and your partner agree on the latter, select the **Specify additional signature requirements** and **Require signed SAML Assertions** options on this tab. When the latter is selected, only the assertion portion of the SAML response message is signed, not the entire SAML response message. This is the only option that appears for SAML 1.x and WS-Federation connections.

SAML 2.0 authentication requests from the service provider (SP) can also be signed to enforce security. This option appears only for SAML 2.0 connections and when the SP-initiated SSO profile is enabled on the **SAML Profiles** tab.

Select **Always Sign Artifact Response** if you want the SAML ArtifactResponse to be signed regardless of the protocol being used to transport it.

Steps

- To continue, select the options based on your partner agreement.

Result

If you are editing an existing connection, you can reconfigure the digital signature policy, which might require additional configuration changes in subsequent tasks.

Specifying XML encryption policy (for SAML 2.0)

For SAML 2.0 configurations, in addition to using signed assertions to ensure authenticity, you and your partner can also agree to encrypt all or part of an assertion to improve privacy.

About this task

You can configure these settings on the **Encryption Policy** tab.

> **Note**
>
> For WS-Fed connections with SAML 2.0 assertions, you cannot encrypt the entire assertion.

Option	Name identifier (SAML_SUBJECT)	Other attributes	Encrypt the SAML_SUBJECT in SLO messages to the IdP	Allow encrypted SAML_SUBJECT in SLO messages from the IdP
None	No encryption.	No encryption.	No encryption.	No encryption.
The entire assertion	Encryption allowed.	Encryption allowed.	Encryption allowed as an available option.	Encryption allowed as an available option.
SAML_SUBJECT (Name Identifier)	Encryption allowed.	Encryption allowed as an available option.	Encryption allowed as an available option.	Encryption allowed as an available option.
One or more attributes	Encryption allowed.	Encryption allowed as an available option.	Encryption allowed as an available option only if you select to allow the entire assertion or the SAML_SUBJECT to be encrypted.	Encryption allowed as an available option only if you select to allow the entire assertion or the SAML_SUBJECT to be encrypted.

> **Note**
>
> To disable the decryption of `EncryptedID` elements when enclosed in a SAML attribute, set the `DecryptEncryptedIdInAttribute` property to `false` in the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.profiles.sp.HandleAuthnResponse.xml` file.

To enable encryption:

Steps

1. Click the **Allow encrypted SAML Assertions and SLO messages** option.
2. Choose whether this identity provider (IdP) partner will encrypt the entire assertion, the `SAML_SUBJECT` name identifier, one or more other attributes, or some combination.
3. If your partner is encrypting the name identifier, indicate whether you will encrypt this attribute in outbound SAML 2.0 single logout (SLO) messages, allow its encryption for inbound messages, or both.

Result

If you are editing an existing connection, you can reconfigure the XML encryption policy, which might require additional configuration changes in subsequent tasks.

Reviewing protocol settings for SP browser SSO

You should confirm your protocol settings for browser-based single sign-on (SSO) before completing configuration.

Steps

1.
    - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
    - To keep your changes, click **Done** and continue with the rest of the configuration.

      > **Tip**
      >
      > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

    - To discard your changes, click **Cancel**.

Reviewing Browser SSO settings

You can review your browser single sign-on (SSO) settings before completing configuration.

Steps

1.
    - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
    - To keep your changes, click **Done** and continue with the rest of the configuration.

      > **Tip**
      >
      > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

  - To discard your changes, click **Cancel**.

Manage the Attribute Query profile in an IdP connection

At the attribute query step, you configure your connection to request user attributes from your partner identity provider (IdP), if you chose this option.

For more information on IdP connection options, see Choosing IdP connection options. Attribute queries do not depend on single sign-on (SSO), but can be used independently or in conjunction with browser SSO or provisioning to provide flexibility in how a user authenticates with service provider (SP) applications. For more information, see Attribute Query and XASP and the /sp/startAttributeQuery.ping SP application endpoint.

Setting the Attribute Authority Service URL

The attribute authority service URL corresponds to the endpoint location where your identity provider (IdP) provider receives attribute query requests.

About this task

Attribute authority is the term used to refer to an IdP that provides user attributes to an attribute requester or your service provider (SP) site. For more information, see Attribute Query and XASP.

Steps

- Enter the fully qualified URL or a relative path if you have defined a base URL on the **General Info** tab. For more information, see Identifying the partner.

Mapping attribute names for Attribute Query

If the application at your site uses different names for user attributes than the names defined by the attribute authority, you must map them on the **Attribute Name Mapping** tab.

About this task

When the service provider (SP) receives a request from a local application to send an attribute query to this attribute authority partner, the requested user attributes are replaced with the names mapped here.

You must predetermine this information in your agreement with this connection partner.

Steps

1. To map an attribute, configure the **Local Name** and **Remote Name** fields for the attribute, then click **Add**.

Choice	Action
Modify an attribute name	Click **Edit** under **Action** for the attribute, make the change, then click **Update**. ⓘ **Note** If you change your mind, ensure that you click **Cancel** under **Actions** and not the **Cancel** button, which discards any other changes you might have made in the configuration steps.
Delete an attribute	Click **Delete** under **Action** for the attribute.

Configuring security policy for Attribute Query

The **Security Policy** tab allows you to specify the digital signing and encryption policy to which you and your partner have agreed.

About this task

These selections will trigger requirements for setting up credentials. For more information, see Configuring security credentials.

This tab also allows you to mask incoming attribute values in log files. For more information, see Attribute masking. When you enable this selection, all user attributes returned from this identity provider (IdP) are masked.

Steps

- Select or clear the check boxes the relevant check boxes.

Reviewing the Attribute Query settings

You can review your attribute query settings before completing configuration.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.
   - To discard your changes, click **Cancel**.

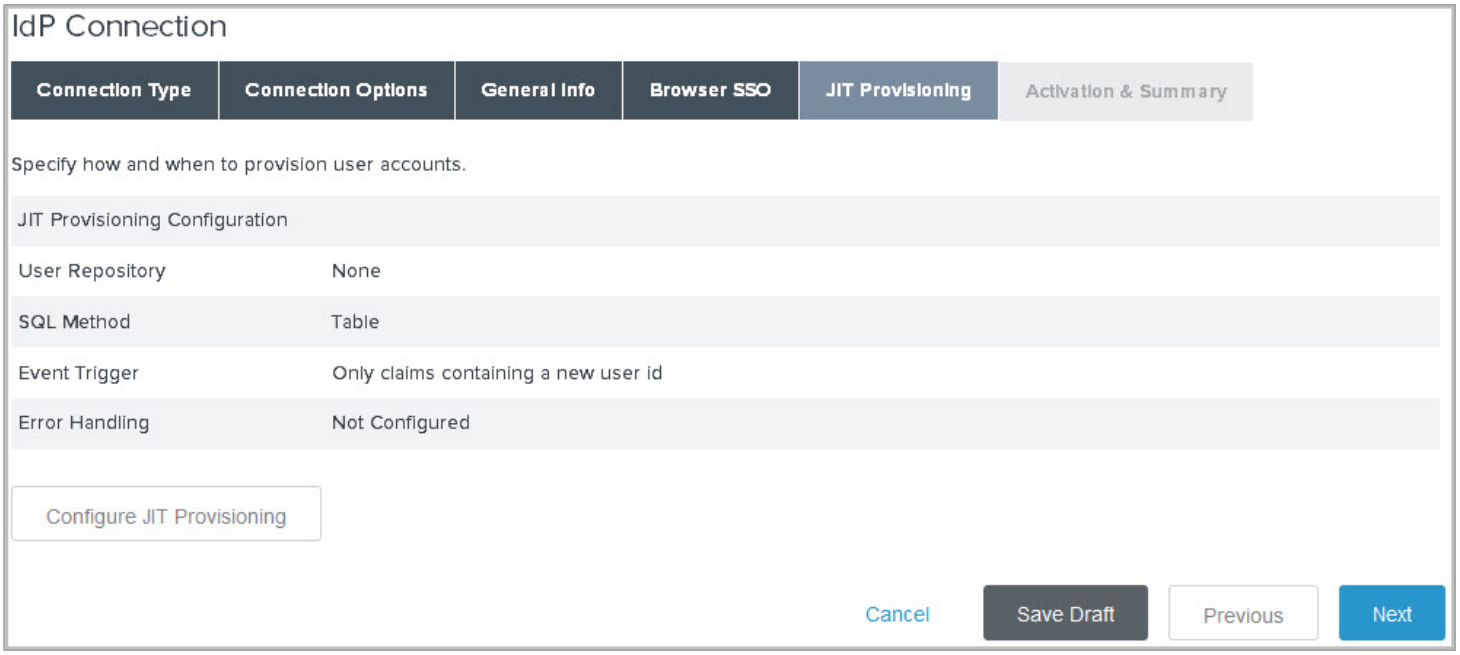
Configuring just-in-time provisioning

PingFederate's just-in-time (JIT) provisioning allows service providers (SPs) to create user accounts on the fly during single sign-on (SSO) events, based on attributes received in SSO tokens from identity providers (IdPs).

About this task

An SP can also use JIT provisioning to update existing user records.

> **Note**
>
> This configuration task is presented in the administrative console only when the **JIT Provisioning** check box is selected on the **Connection Options** tab.



IdP Connection

Specify how and when to provision user accounts.

JIT Provisioning Configuration	
User Repository	None
SQL Method	Table
Event Trigger	Only claims containing a new user id
Error Handling	Not Configured

Steps

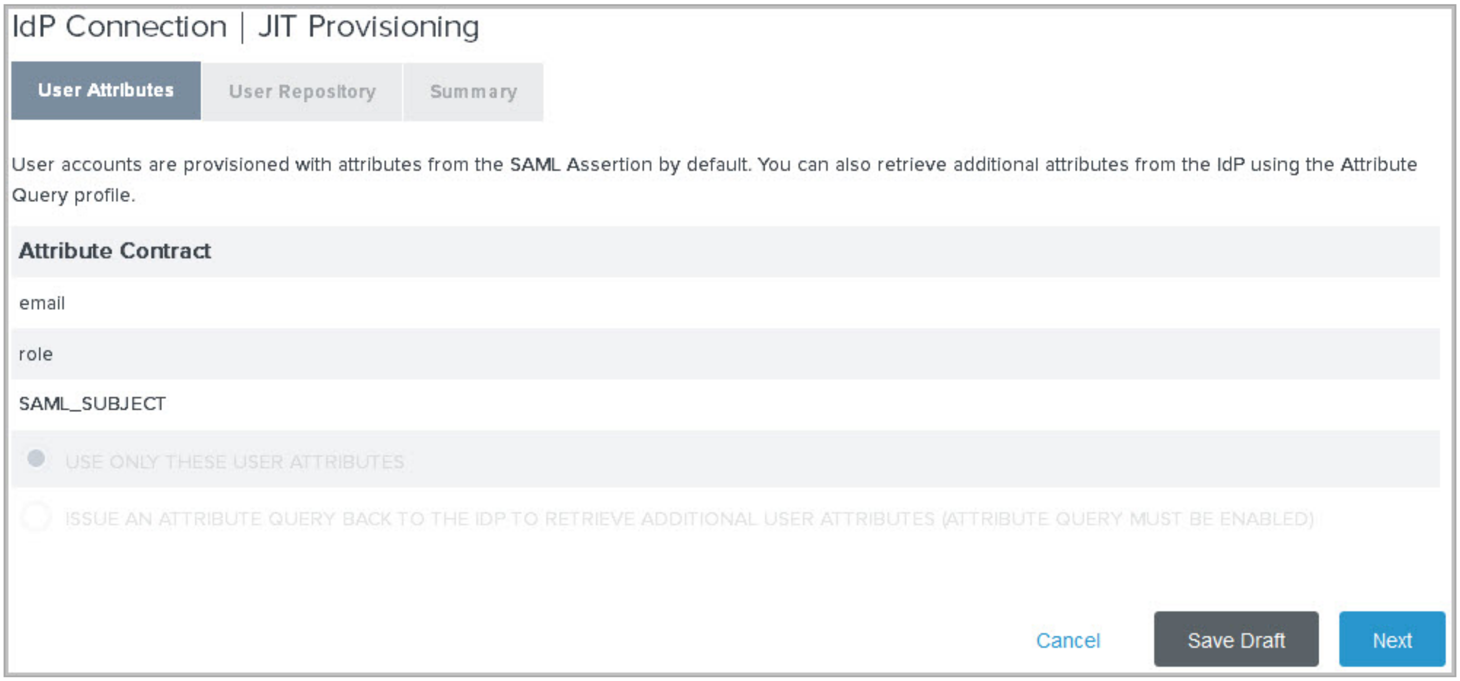
1. Go to **Authentication** > **Integration** > **IdP Connections**.
2. Create a new IdP connection or select an existing IdP connection .
3. On the **Connection Type** tab, select the **Browser SSO Profiles** check box and a protocol from the list.
4. On the **Connection Options** tab, select the **Browser SSO** check box and then the **JIT Provisioning** check box.
5. Complete the **Browser SSO** configuration.
6. On the **JIT Provisioning** tab, click **Configure User Provisioning** to begin the configuration of JIT provisioning.

Selecting attribute sources (SAML 2.0)

For SAML 2.0 connections, the server can be configured to use only assertion attributes for user provisioning, or to retrieve more attributes from the identity provider (IdP) in a follow-on attribute query transaction.

About this task

The **User Attributes** tab displays the attributes expected in the assertion from this IdP.



IdP Connection | JIT Provisioning

User accounts are provisioned with attributes from the SAML Assertion by default. You can also retrieve additional attributes from the IdP using the Attribute Query profile.

Attribute Contract
email
role
SAML_SUBJECT

● USE ONLY THESE USER ATTRIBUTES

○ ISSUE AN ATTRIBUTE QUERY BACK TO THE IDP TO RETRIEVE ADDITIONAL USER ATTRIBUTES (ATTRIBUTE QUERY MUST BE ENABLED)

> **Note**
>
> The attribute query is a SAML 2.0 profile. For OpenID Connect, SAML 1.x, and WS-Federation connections, this tab is not presented. PingFederate uses only attributes from the assertion for user provisioning.

Steps

- If you and your IdP partner have agreed to use the Attribute Query profile for provisioning, select that option before leaving this tab.

  You configure the attribute query profile later in the task flow.

Identifying the user repository

PingFederate's just-in-time (JIT) provisioning supports several directory servers and Microsoft SQL Server.

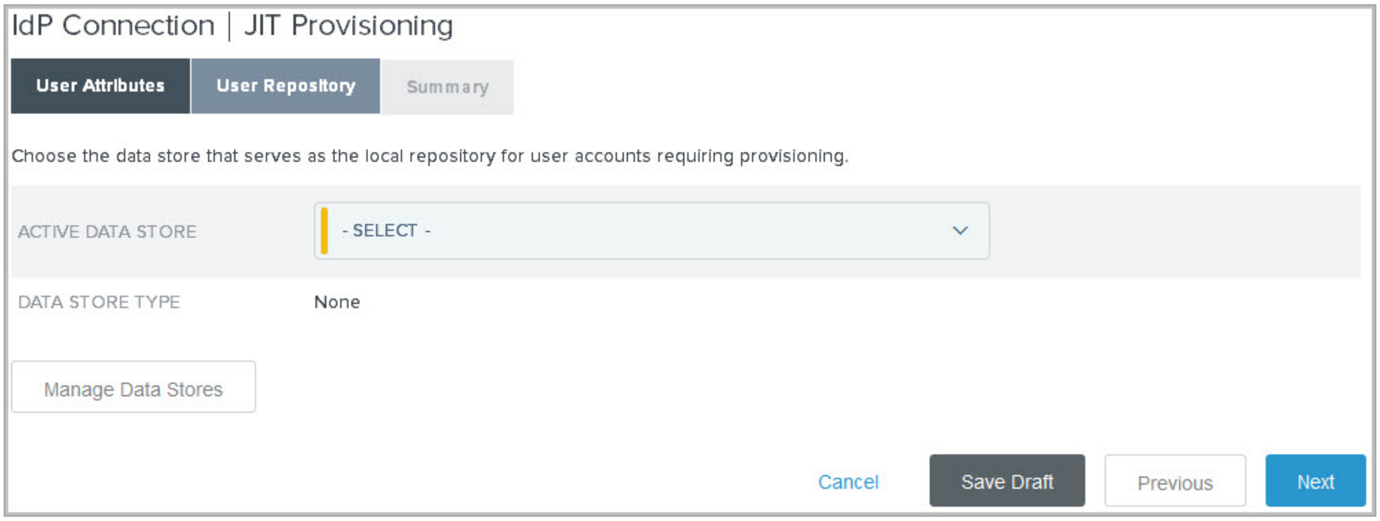
About this task

For more information on versions and requirements supported by PingFederate, see System requirements.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers.

Steps

- On the **User Repository** tab, select a datastore from the **Active Data Store** list.


IdP Connection | JIT Provisioning

Choose the data store that serves as the local repository for user accounts requiring provisioning.

ACTIVE DATA STORE - SELECT -

DATA STORE TYPE None

Tip

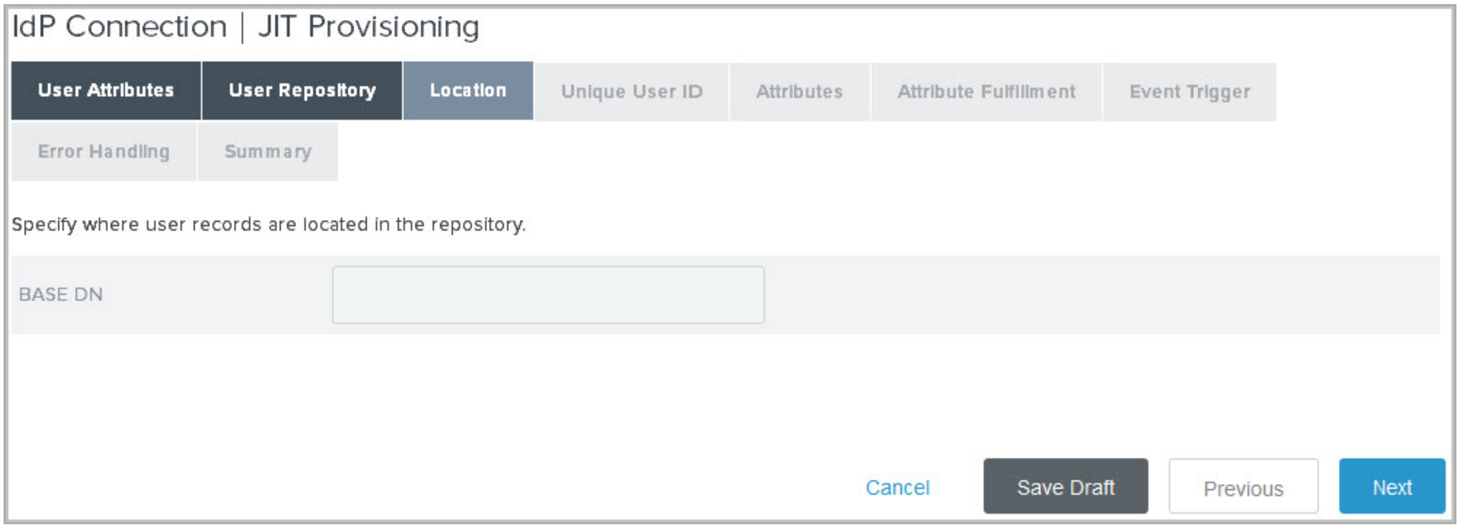
If the desired datastore is not shown in the list, PingFederate has not been configured to access it. Click **Manage Data Stores** to configure your datastore.

- If you are using an LDAP store, see the sections immediately following:
  - Specifying an LDAP user-record location
  - Entering an LDAP filter
  - Identifying provisioning attributes for LDAP
- If you are using a Microsoft SQL Server, skip to this section:
  - Choose a SQL method

Specifying an LDAP user-record location

After choosing a datastore, indicate where in the store PingFederate should write new user records or update existing ones.

About this task



Specify where user records are located in the repository.

BASE DN

> **Note**
>
> The **Location** tab appears only when an LDAP datastore is chosen on the **User Repository** tab.

Steps

1. Enter the base distinguished name (DN) in the **Base DN** field.

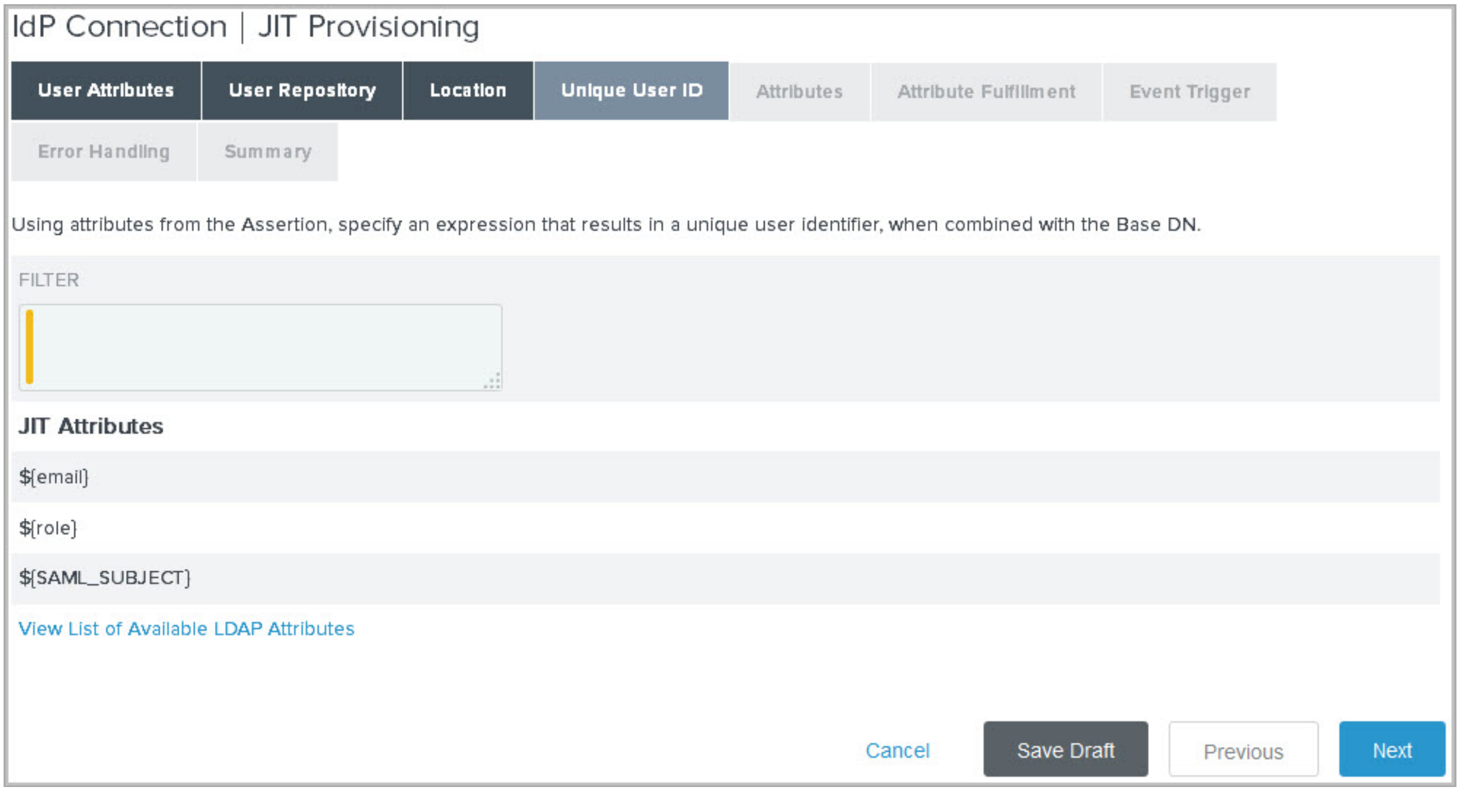
   A base DN is the DN of the tree structure in which the search begins. Leave this field blank if records are located at the LDAP root.

Entering an LDAP filter

On the **Unique User ID** tab, create an LDAP filter to identify user accounts to be provisioned or updated during single sign-on (SSO) events.

About this task

PingFederate uses this expression in conjunction with the `Base DN` value defined on the **Location** tab to locate existing account records and to add new ones.



IdP Connection | JIT Provisioning

Using attributes from the Assertion, specify an expression that results in a unique user identifier, when combined with the Base DN.

FILTER

JIT Attributes

${email}

${role}

${SAML_SUBJECT}

Note

This tab appears only when an LDAP datastore is chosen on the **User Repository** tab.

Steps

- Enter the statement in the **Filter** field.

  The filter is in the form: `attribute=${value}` .

  Note

  Unlike filters used to retrieve LDAP attributes for adapter mapping, do not enclose the statement in parentheses.

  The left-side variable is an attribute in your user-datastore. Click the link near the lower-left corner of the tab to see a list of available attributes.

  The right side of the filter uses one or more attribute values passed in from the SSO token. Variables for these attributes, including the correct syntax, are listed under **JIT Attributes**.

  Tip

  You can reference attribute values in the form of `$\{attributeName:-defaultValue}` . The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

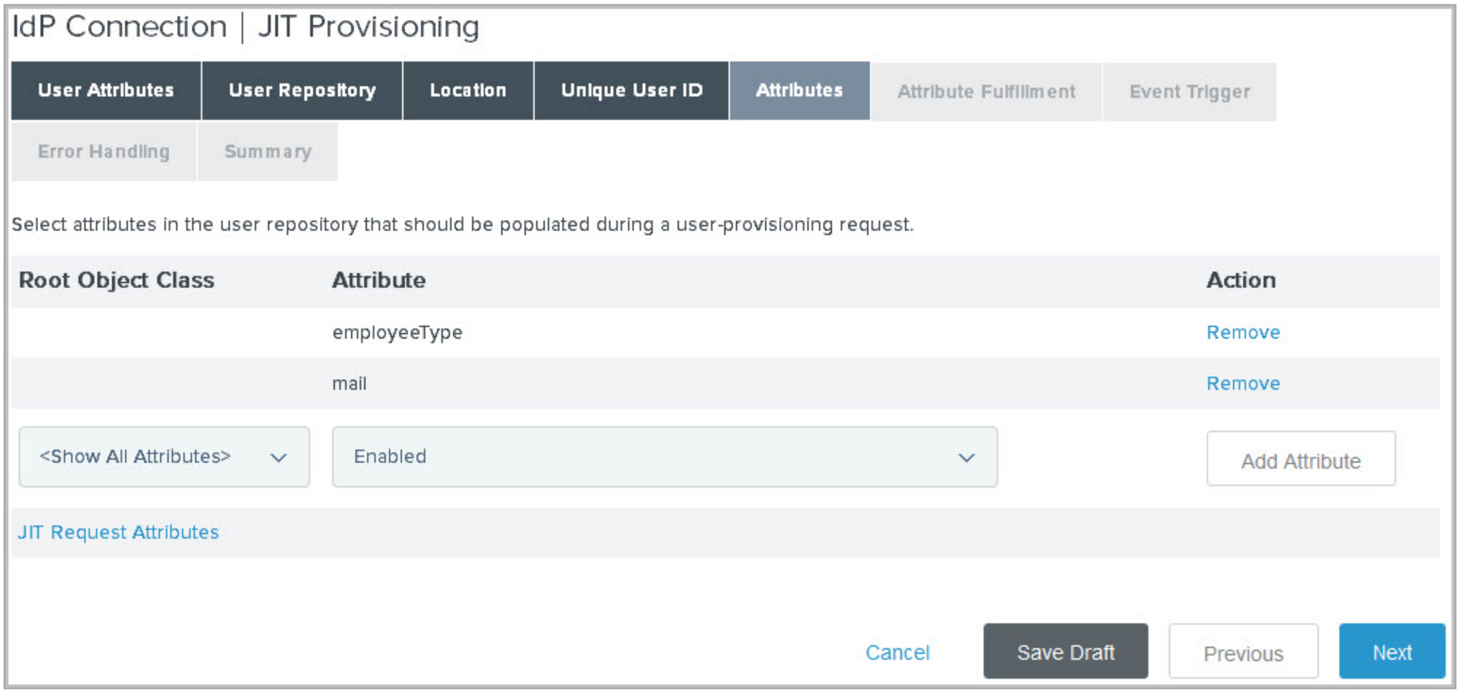
Tip

If you are unfamiliar with writing LDAP queries, see the documentation accompanying your LDAP installation.

Identifying provisioning attributes for LDAP

On the **Attributes** tab, you can select the datastore attributes to be provisioned.

About this task



IdP Connection | JIT Provisioning

Select attributes in the user repository that should be populated during a user-provisioning request.

Root Object Class	Attribute	Action
	employeeType	Remove
	mail	Remove
<Show All Attributes>	Enabled	Add Attribute

Note

This tab appears only when an LDAP datastore is chosen on the **User Repository** tab.

Steps

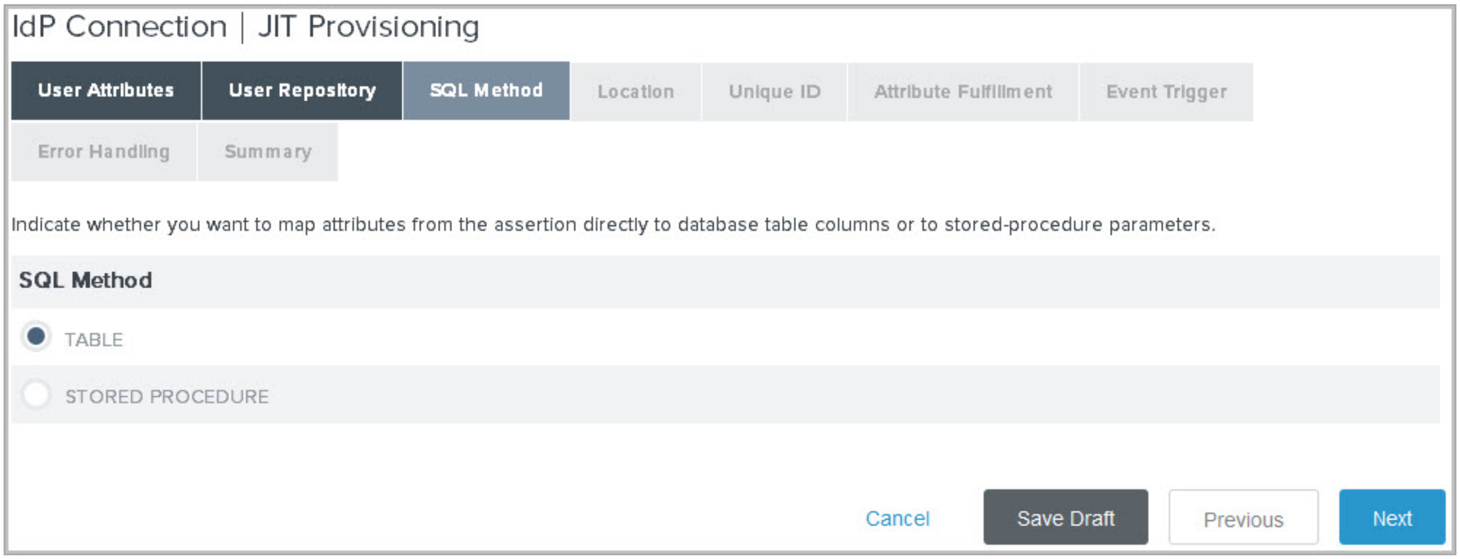
1. Select a root object class and an attribute from the lists, and then click **Add Attribute**.
2. Repeat for each attribute requiring provisioning.

Choosing a SQL method

PingFederate allows you to map attributes directly to a single database table, the default, or to SQL stored-procedure parameters for Java Database Connectivity (JDBC) datastores,

About this task

Choose and configure the preferred method on the **SQL Method** tab.


IdP Connection | JIT Provisioning

Indicate whether you want to map attributes from the assertion directly to database table columns or to stored-procedure parameters.

SQL Method

● TABLE

○ STORED PROCEDURE

Note

This tab appears only when you specify Microsoft SQL Server datastore on the **User Repository** tab.

Steps

- Make a selection as needed and click **Next**.

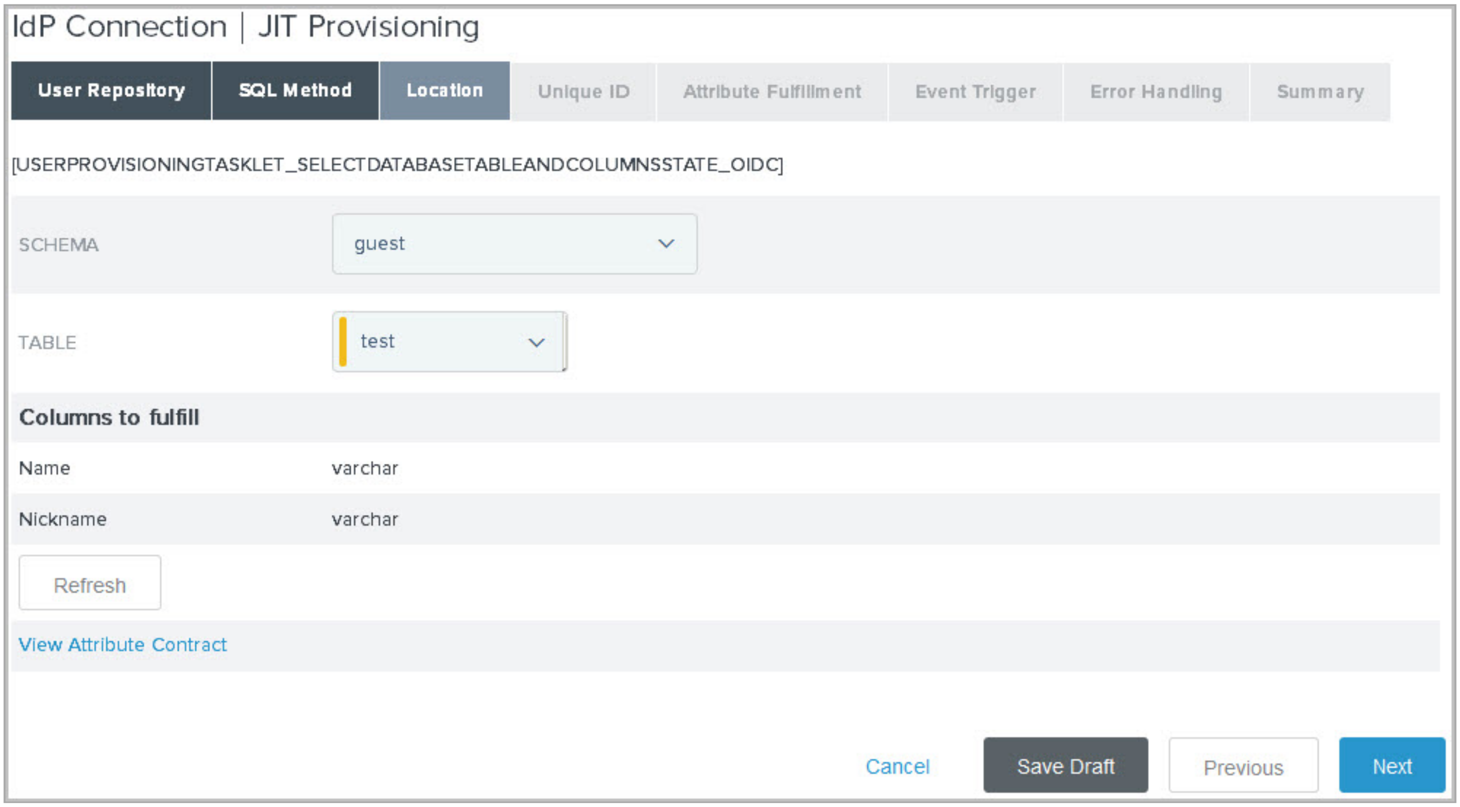
  Depending on the selection, different steps appear under the **JIT Provisioning** task. See the sections indicated for more information.
- If mapping attributes directly to a table, see the topics sections immediately following:
  - Specifying a database user-record location
  - Specifying a unique ID database column
- If using a stored procedure, skip to Specifying a stored procedure location

Specifying a database user-record location

For database provisioning to a table, you must indicate where PingFederate should write new user records or update existing ones.

About this task

Configure the location settings on the **Location** tab.



IdP Connection | JIT Provisioning

[USERPROVISIONINGTASKLET_SELECTDATABASETABLEANDCOLUMNSSTATE_OIDC]

SCHEMA: guest

TABLE: test

Columns to fulfill

Name	varchar
Nickname	varchar

Note

This tab appears only when you choose a Microsoft SQL Server datastore on the **User Repository** tab and select the **Table** option on the **SQL Method** tab.

Steps

- On the **Location** tab, select the database schema and the table.

Field	Description
Schema	Select the table structures that store information within the database.
Table	Select the name of the database table that contains user records.
Columns to fulfill	For the selected table, all attributes and their data types are displayed. If the list of columns is not current due to any recent changes, click **Refresh**. On the **Attribute Fulfillment** tab, all attributes must be mapped for the database insertion to succeed, although a null entry can be used for optional attributes.

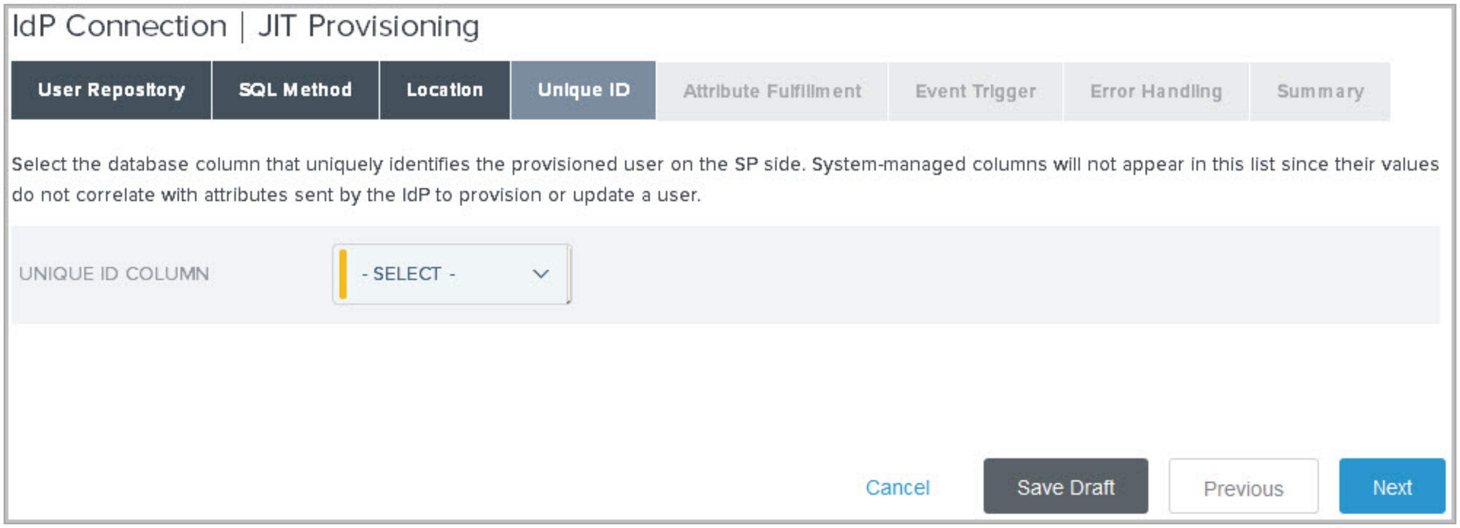
Tip

Click the link near the lower-left corner of the tab to see a list of available attributes from the single sign-on (SSO) token.

Specifying a unique ID database column

PingFederate uses the database column you specify on the **Unique ID** tab to check whether a user record already exists for the incoming single sign-on (SSO) token.

About this task



IdP Connection | JIT Provisioning

Select the database column that uniquely identifies the provisioned user on the SP side. System-managed columns will not appear in this list since their values do not correlate with attributes sent by the IdP to provision or update a user.

UNIQUE ID COLUMN - SELECT -

> **Note**
>
> This tab only appears if you have chosen a Microsoft SQL Server datastore on the **User Repository** tab and you selected the **Table** option on the **SQL Method** tab.

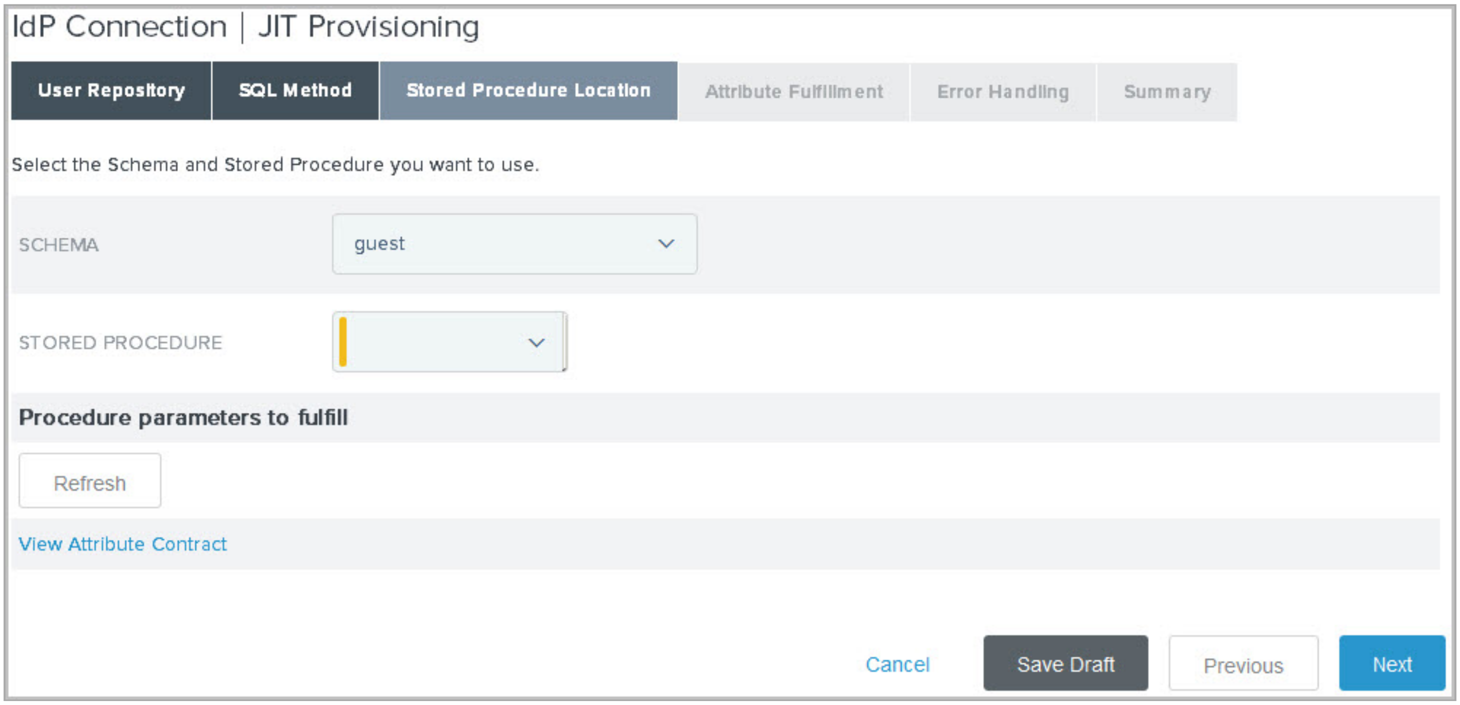
Steps

- Select a column that represents a unique characteristic about the database entry for a particular user.

Specifying a stored procedure location

If you are using a stored procedure for provisioning the user database, specify its location on the **Stored Procedure Location** tab.

About this task



IdP Connection | JIT Provisioning

Select the Schema and Stored Procedure you want to use.

SCHEMA: guest

STORED PROCEDURE:

Procedure parameters to fulfill

Note

This window appears only when you have chosen a Microsoft SQL Server datastore on the **User Repository** tab and the you have selected the **Stored Procedure** option on the **SQL Method** tab.

Steps

1. From the **Schema** menu, select the table structure that contains stored procedure within the database.
2. From the **Stored Procedure** menu, select the stored procedure needed to provision the user database.

   Important

   The database account used by PingFederate must have access to the schema in which the stored procedure is located and execute permission for the procedure.

   Result:

   For the selected procedure, all parameters and their data types display in the **Procedure parameters to fulfill** section.

   Note

   To see a list of available attributes from the single sign-on (SSO) token, click **View Attribute Contract**. You map attribute values from the SSO token to the parameters on the **Attribute Fulfillment** tab.

   Troubleshooting:

   If the list of parameters is not current due to any recent procedure revisions, click **Refresh**.

Mapping attributes to a user account

Map incoming attributes to the account attributes on an LDAP server, the columns in a database table on a Microsoft SQL Server, or the parameters of a Microsoft SQL Server stored procedure.

About this task

In addition to values obtained from the single sign-on (SSO) token, you can map attributes from the context of the SSO token text, with or without reference values from the SSO token, and expression if enabled.

If you select a Microsoft SQL Server datastore on the **User Repository** tab, then on the **Attribute Fulfillment** tab you can test the insertion of attribute values into the database table or the stored procedure. When mapping to a database column of the `datetime` or `smalldatetime` data type, if you are not using a stored procedure to convert the incoming string value, you can use a PingFederate Java conversion method through OGNL expressions.

Steps

1. On the **Attribute Fulfillment** tab, select a source from the list for each target attribute or parameter.

   Choose from:

   - `Assertion or Provider Claims`

     Values are contained in the SSO token from this identity provider (IdP). When you select this, the associated **Value** list is populated by the attribute contract.

   - `Context`

     Values are returned from the context of the transaction at runtime.

     > **Note**
     >
     > As the HTTP Request is retrieved as a Java object rather than text, OGNL expressions are more appropriate to evaluate and return values. Choose **Expression**from the list and then click **Edit** to enter an expression.

   - `Attribute Query`

     This choice appears only if you choose the **Attribute Query** profile for provisioning.

     To map an attribute-query value, use the syntax `${query_attribute}`. You can combine attribute-query values with references to attributes in the attribute contract; for example, `${query_attribute}+$\{attribute.`

     References to attributes not contained in the attribute contract result in an attribute query back to the IdP partner.

   - `Expression`

     > **Tip**
     >
     > Enable OGNL expression by editing the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.ExpressionManager.xml` file. Restart PingFederate after saving the change.
     > For a clustered PingFederate environment, edit the `org.sourceid.common.ExpressionManager.xml` file on the console node, sign on to the administrative console to replicate this change to all engine nodes in the **System** > **Server** > **Cluster Management** window, and restart all nodes.

This option provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

> **Tip**
>
> If you need to map multiple attribute values from one or multiple sources to one attribute value, use an OGNL expression to create it.
> For database mapping, if the data type of a target parameter is `datetime` or `smalldatetime`, you can use an expression to convert date-time strings from the SSO token. After selecting **Expression**, click **Datetime OGNL Examples** for syntax information and examples.

- `System Managed`

  This mapping option appears only when any automatically assigned attributes are among columns to be provisioned, such as an identity or a timestamp column on the Microsoft SQL Server.

- `Text`

  The value is what you enter. This can be text only, or you can mix text with references to any of the values from the SSO token, using the `${attribute}` syntax.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

> **Note**
>
> For LDAP mapping, choose **Text** as the **Source** for the `objectClass` attribute.
> For mapping into a database, if no entry is required for a column, you can leave the field blank. A blank entry results in an empty string in the database for string data types and null for all other data types. Alternatively, for string types, you can enter `null` in the field to explicitly set `null` in the column.

2. Select or enter an attribute value.

   All values must be mapped. For optional table columns, you can leave the field blank or, for string data types, enter `null` to avoid empty strings.

   No value is required for **System Managed** attributes.

> **Note**
>
> For **Active Directory**, enter `user` in the **objectClass** field. For Oracle Directory Server or Oracle Unified Directory, enter `inetOrgPerson`.

3. **Optional:** When mapping to a Microsoft SQL Server datastore, test the insertion.

   Choose from:

   - If testing from a table:

     1. Click **Test insert into <*table*>**.

2. Enter values for each applicable target parameter.

3. Click **Test Insert**.

If the test succeeds, a confirmation displays along with the values inserted.

> **Caution**
>
> Unless you want to keep the test values in the database, click **Roll Back All Test Inserts**.

- If testing from a stored procedure:

1. Click **Test call to <*procedure*>**.

2. Enter values for each applicable target parameter.

3. Click **Test Stored Procedure Call**.

For stored procedures, only a confirmation displays if the test is successful, indicating that the procedure was populated with parameter values.

> **Caution**
>
> No roll back feature is provided because PingFederate does not know the result of the procedure. Database rollback must be handled manually.

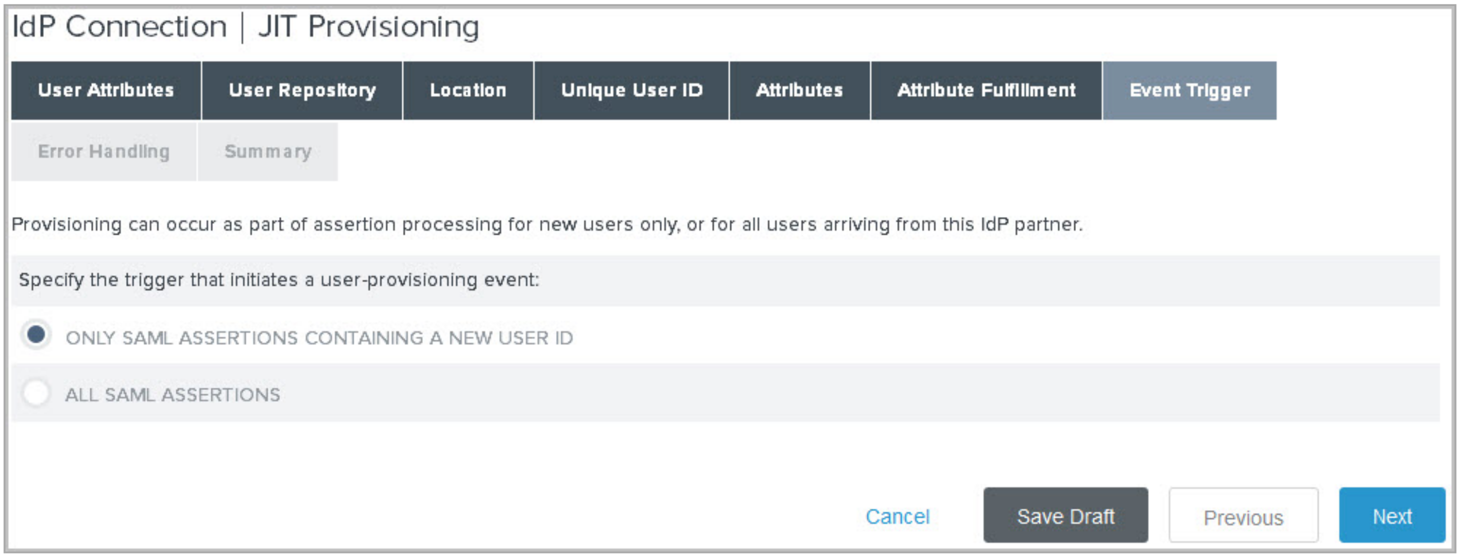
When finished, click **Return to Attribute Fulfillment**.

Choosing an event trigger

Choose whether PingFederate initiates user provisioning only when the user identifier is new, or every time your site receives a single sign-on (SSO) token.

About this task

If you choose to have PingFederate initiate user provisioning every time your site receives an SSO token, for all SSO tokens, an existing user account is always updated with incoming attributes.



IdP Connection | JIT Provisioning

Provisioning can occur as part of assertion processing for new users only, or for all users arriving from this IdP partner.

Specify the trigger that initiates a user-provisioning event:

● ONLY SAML ASSERTIONS CONTAINING A NEW USER ID

○ ALL SAML ASSERTIONS

> **Note**
>
> This tab does not appear for a Microsoft SQL Server datastore if provisioning is accomplished using a stored procedure, because the procedure is always called for all SSO tokens. The procedure should handle both provisioning new users and updating existing ones.

Steps

- On the **Event Trigger** tab, in the **Specify the trigger that initiates a user-provisioning event** section, select one of the following:

  Choose from:

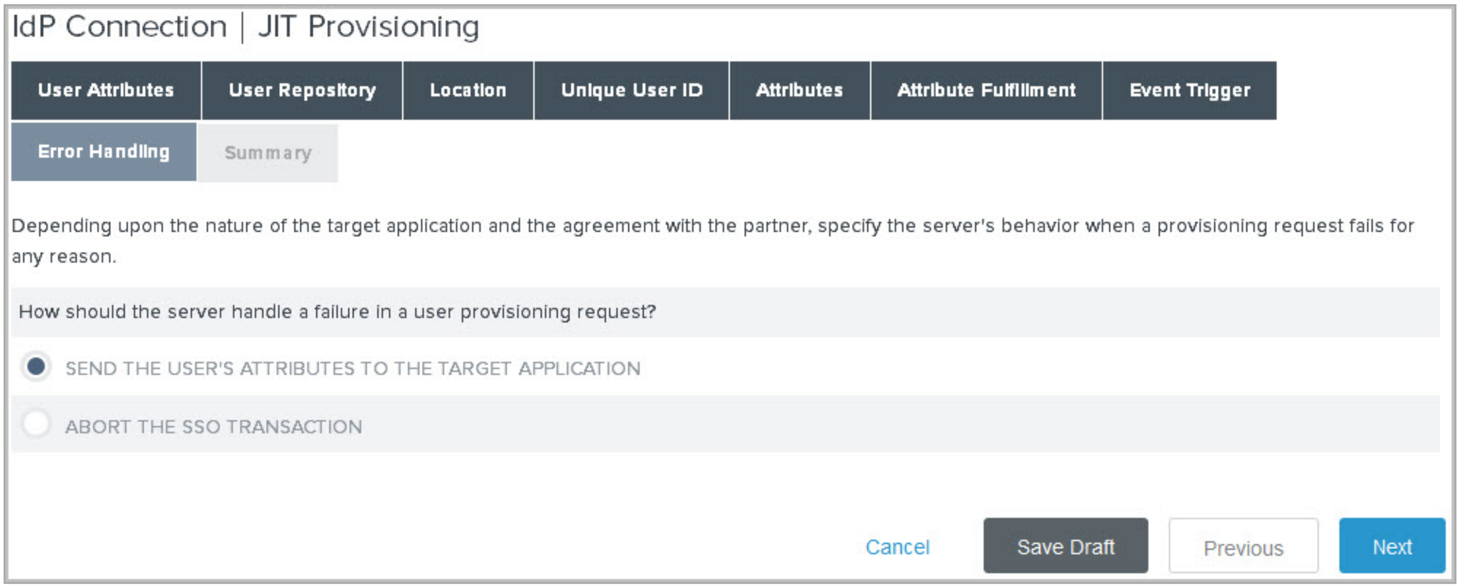
  - **Only SAML Assertations Containing a New User ID**
  - **All SAML Assertations**

Configuring an error handling method

If user provisioning fails for any reason during single sign-on (SSO) events, you can choose to continue the process by passing the user's attributes to your target application or to abort the SSO transaction.

About this task

When SSO is aborted, the user is redirected to an error page and the failure is written to the server log.



Depending upon the nature of the target application and the agreement with the partner, specify the server's behavior when a provisioning request fails for any reason.

How should the server handle a failure in a user provisioning request?

● SEND THE USER'S ATTRIBUTES TO THE TARGET APPLICATION

○ ABORT THE SSO TRANSACTION

Steps

- On the **Error Handling** tab, in the **How should the server handle a failure in a user provisioning request?** section, select one of the following:

  Choose from:

  - **Send the User's Attributes to the Target Application**
  - **Abort the SSO Transaction**

Reviewing the JIT provisioning configuration

You can review your just-in-time (JIT) provisioning settings before completing configuration.

Steps

1. - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Configuring SCIM inbound provisioning

In the **IdP Connections** window, configure System for Cross-domain Identity Management (SCIM) inbound provisioning to provide a two-way mapping of attributes.

About this task

The first way facilitates SCIM operations used to create and update records in the datastore.

The second way allows the same SCIM client to retrieve those records and have the attribute values mapped back to their corresponding designation in the client store.

The dual mapping provides greater flexibility, especially for OGNL-expression transformations (for example, converting two attributes into one multivalued attribute and then back again).

Learn more in Writing user information to the datastore and Configuring a SCIM response.

> **Note**
>
> SCIM-client requests must include authentication credentials, which you configure on the **Credentials > Back-Channel Authentication** tab. The same credentials needed for single sign-on (SSO), are also used for SCIM transactions.

Steps

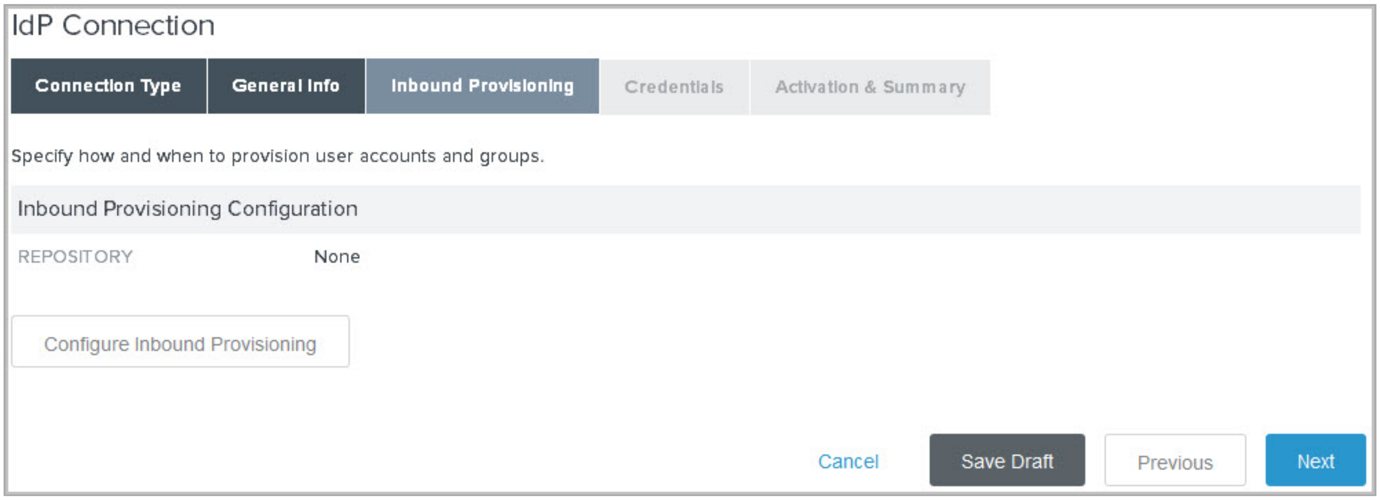
1. On the **Authentication > Integration > IdP Connections** window, create a new IdP connection or select an existing IdP connection.

2. On the **Connection Type** tab, select the **Inbound Provisioning** check box and one of the following options:

   Choose from:

   - **User Support**
   - **User and Group Support**

3. On the **Inbound Provisioning** tab, click **Configure Inbound Provisioning** to begin the configuration of SCIM inbound provisioning.



Specify how and when to provision user accounts and groups.

Inbound Provisioning Configuration

REPOSITORY None

Specifying the user repository

PingFederate supports Active Directory (AD) user stores and custom identity store provisioners for inbound provisioning.

About this task

Manage the datastore serving as the local repository on the **Repository** tab.

Note

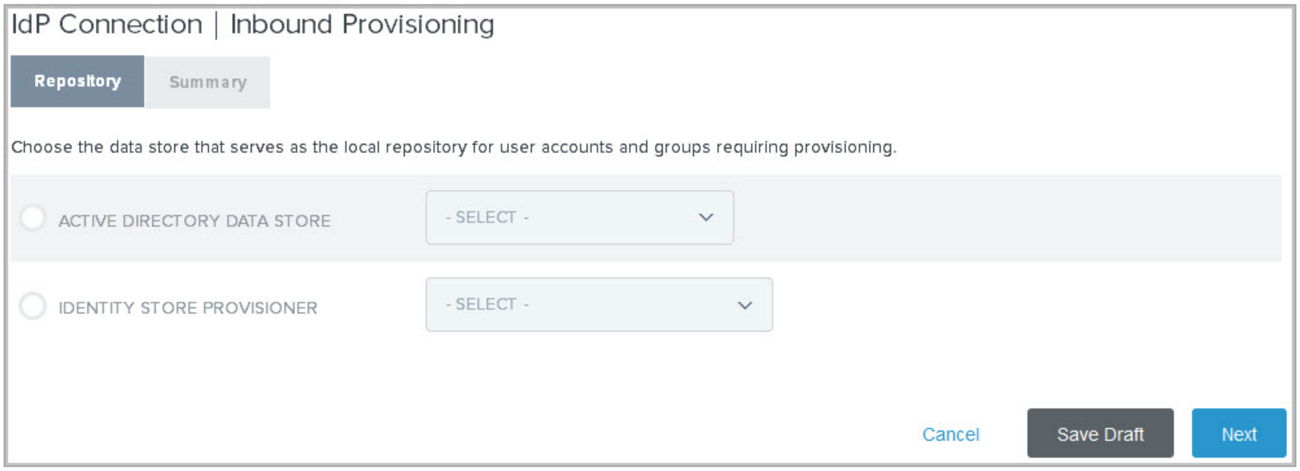
Active Directory user stores require an LDAPS connection to the datastore.

Steps

- Choose one of the following options, and then specify the datastore from the list.

  Choose from:

  - **Active Directory Data Store**
  - **Identity Store Provisioner**



IdP Connection | Inbound Provisioning

Repository | Summary

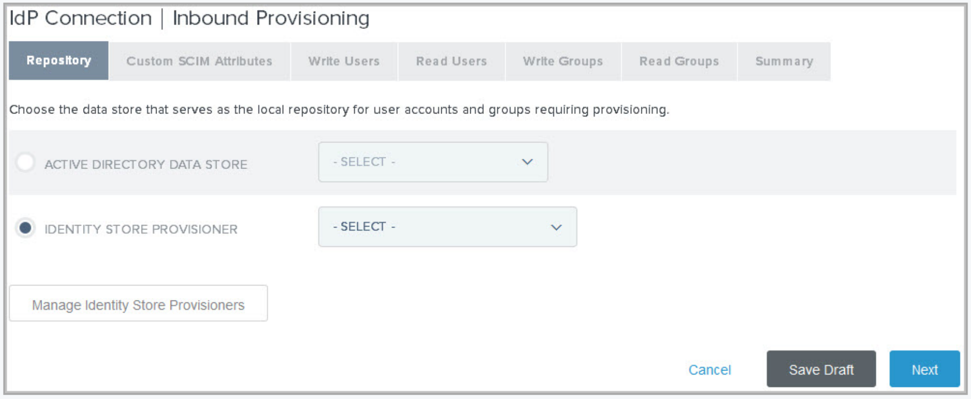
Choose the data store that serves as the local repository for user accounts and groups requiring provisioning.

○ ACTIVE DIRECTORY DATA STORE - SELECT -

○ IDENTITY STORE PROVISIONER - SELECT -

Tip

If the correct datastore is not shown in the list, then PingFederate is not configured to access the store. Click **Manage Data Stores** to set up the desired datastore.



IdP Connection | Inbound Provisioning

Repository | Custom SCIM Attributes | Write Users | Read Users | Write Groups | Read Groups | Summary

Choose the data store that serves as the local repository for user accounts and groups requiring provisioning.

○ ACTIVE DIRECTORY DATA STORE - SELECT -

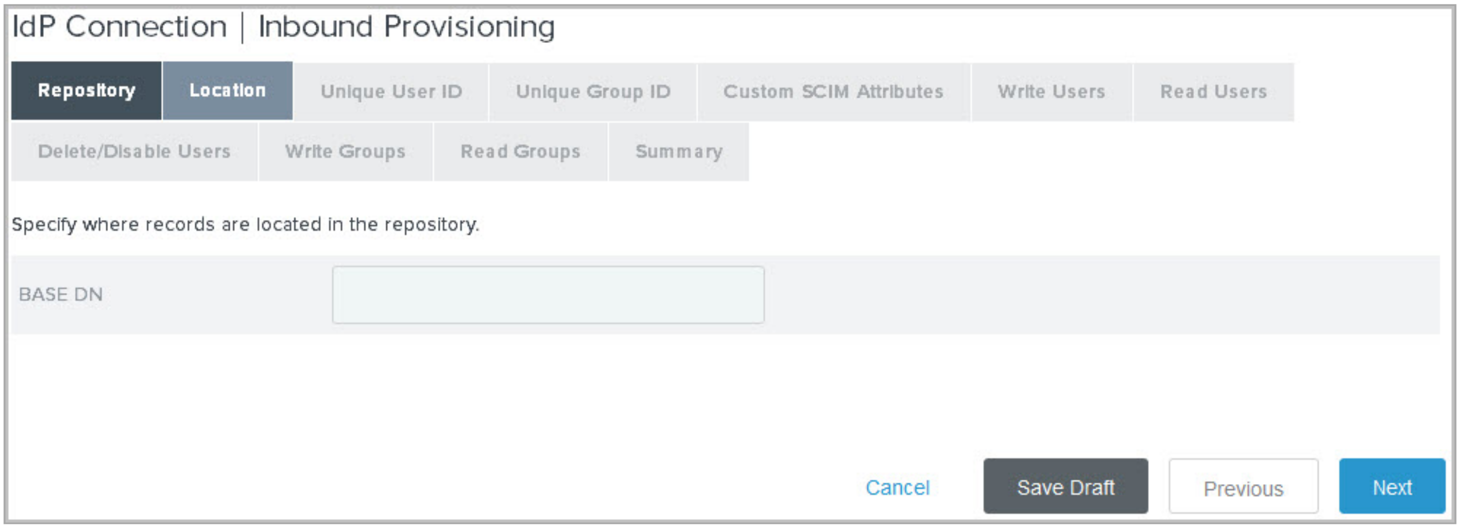
● IDENTITY STORE PROVISIONER - SELECT -

Identifying an LDAP user-record location

After choosing a datastore, you can indicate where in the datastore user and group records exist so PingFederate can create, read, update, or delete or disable them.

About this task

These settings can be configured on the **Location** tab.



IdP Connection | Inbound Provisioning

Specify where records are located in the repository.

BASE DN

Note

This tab appears only if you are configuring an LDAP user store for provisioning.

Steps

- Enter the base distinguished name (DN) of the tree structure where user records are stored in the **Base DN** field.

  Note

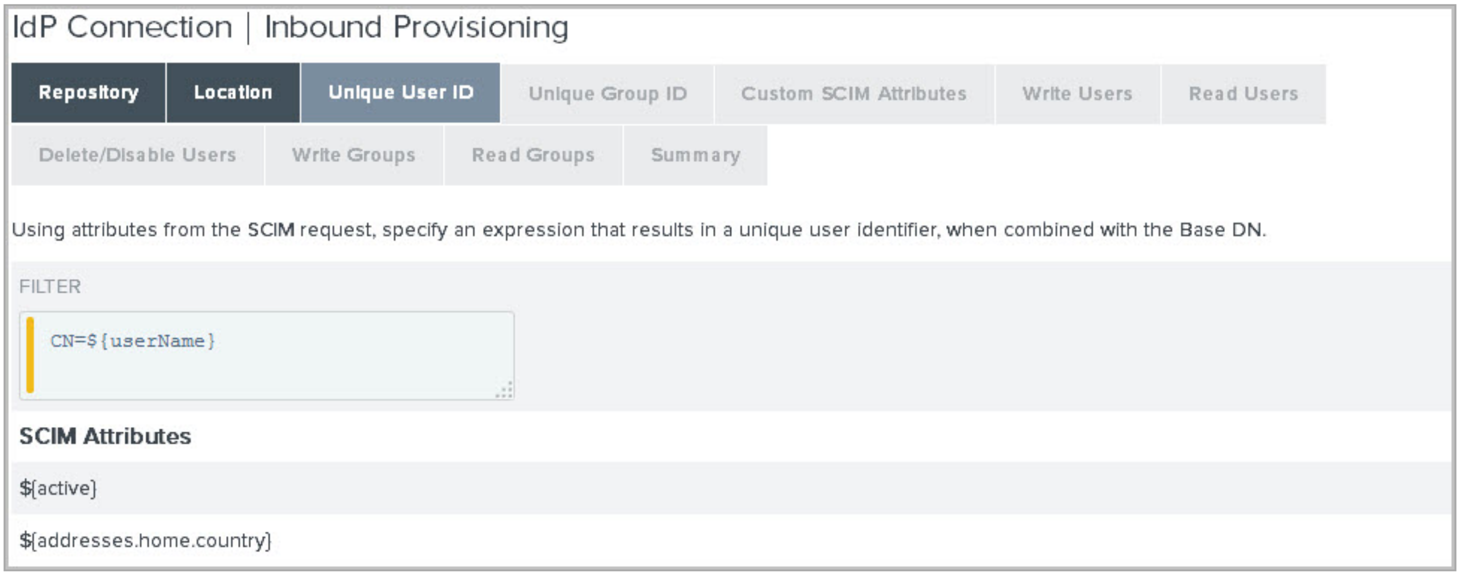
  PingFederate looks only at this node level or below it for user accounts that need provisioning

Defining a unique user ID

On the **Unique user ID** tab, you can create an LDAP filter to resolve user accounts for System for Cross-domain Identity Management (SCIM) operations.

About this task

PingFederate uses LDAP filter in conjunction with the **Base DN** value, defined on the **Location** tab, to add new account records.



Using attributes from the SCIM request, specify an expression that results in a unique user identifier, when combined with the Base DN.

FILTER

CN=${userName}

SCIM Attributes

${active}

${addresses.home.country}

> **Note**
>
> This tab only appears if you are configuring an LDAP user store for provisioning.

Steps

1. Enter the statement in the **Filter** text field.The filter is in the form: `attribute=$\{value}` where `attribute` is an attribute in your user-datastore and `value` is the attribute value or values passed in from the SCIM request. To see a list of available attributes in your user-datastore, click **View List of Available LDAP Attributes**. Variables for these attributes, including the correct syntax, are listed under **SCIM Attributes**.

> **Note**
>
> Unlike filters used to retrieve LDAP attributes for adapter mapping, do not enclose the statement in parentheses.

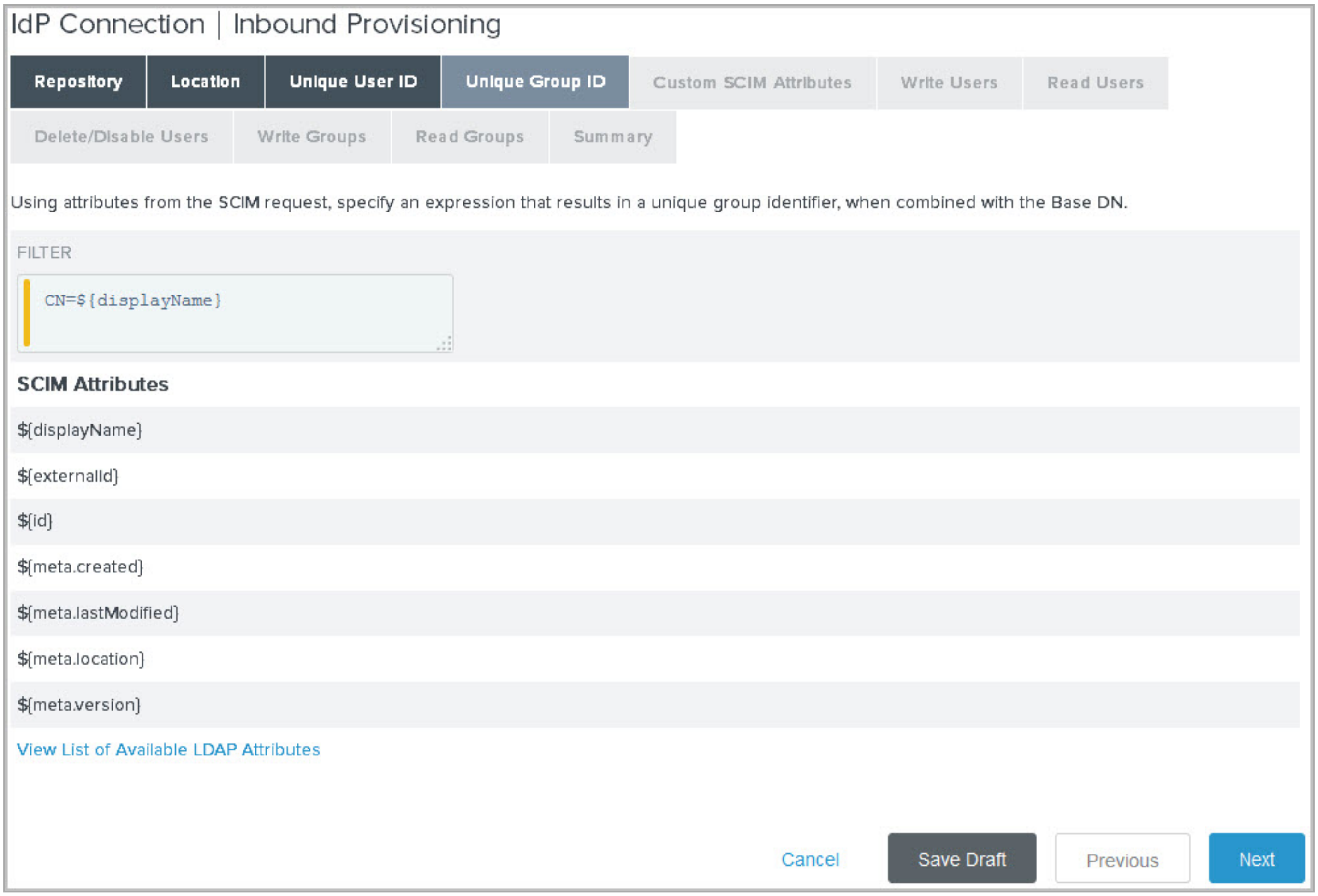
> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value. This is optional.
> If you are unfamiliar with writing LDAP queries, see the documentation accompanying your LDAP installation.

Defining a unique group ID

On the **Unique Group ID** tab, you can create an LDAP filter to resolve groups for System for Cross-domain Identity Management (SCIM) operations.

About this task

PingFederate uses this LDAP filter in conjunction with the **Base DN** value, defined on the **Location** tab, to add new groups.



IdP Connection | Inbound Provisioning

Using attributes from the SCIM request, specify an expression that results in a unique group identifier, when combined with the Base DN.

FILTER

```
CN=${displayName}
```

SCIM Attributes

${displayName}

${externalId}

${id}

${meta.created}

${meta.lastModified}

${meta.location}

${meta.version}

Note

This tab appears only if you are configuring an LDAP user store for provisioning and you have selected the **User and Group Support** option on the **Connection Type** tab.

Steps

1. Enter the statement in the **Filter** text field.The filter is in the form: `attribute=$\{value}` where `attribute` is an attribute in your user-datastore and `value` is the attribute value or values passed in from the SCIM request. To see a list of available attributes in your user-datastore, click **View List of Available LDAP Attributes**. Variables for these attributes, including the correct syntax, are listed under **SCIM Attributes**.

Note

Unlike filters used to retrieve LDAP attributes for adapter mapping, do not enclose the statement in parentheses.

Tip

You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value. This is optional.

If you are unfamiliar with writing LDAP queries, see the documentation accompanying your LDAP installation.

Defining custom SCIM attributes

When configuring System for Cross-domain Identity Management (SCIM) inbound provisioning, you can define custom attributes.

PingFederate supports SCIM attributes in the core schema and custom attributes through a schema extension.

> **Note**
>
> Custom attributes are optional. If your use case does not require any additional attributes, click **Next** on the **Custom SCIM Attributes** tab.

To support custom attributes, you must specify the schema extension and the custom attributes in the connection. There are four attribute types:

- Simple attributes
- Simple multivalued attributes
- Complex attributes
- Complex multivalued attributesThe following fragment illustrates a SCIM message supporting schema extension `urn:scim:schemas:extension:custom:1.0` with four attributes, one of each attribute type. The table afterward describes the details of each attribute.

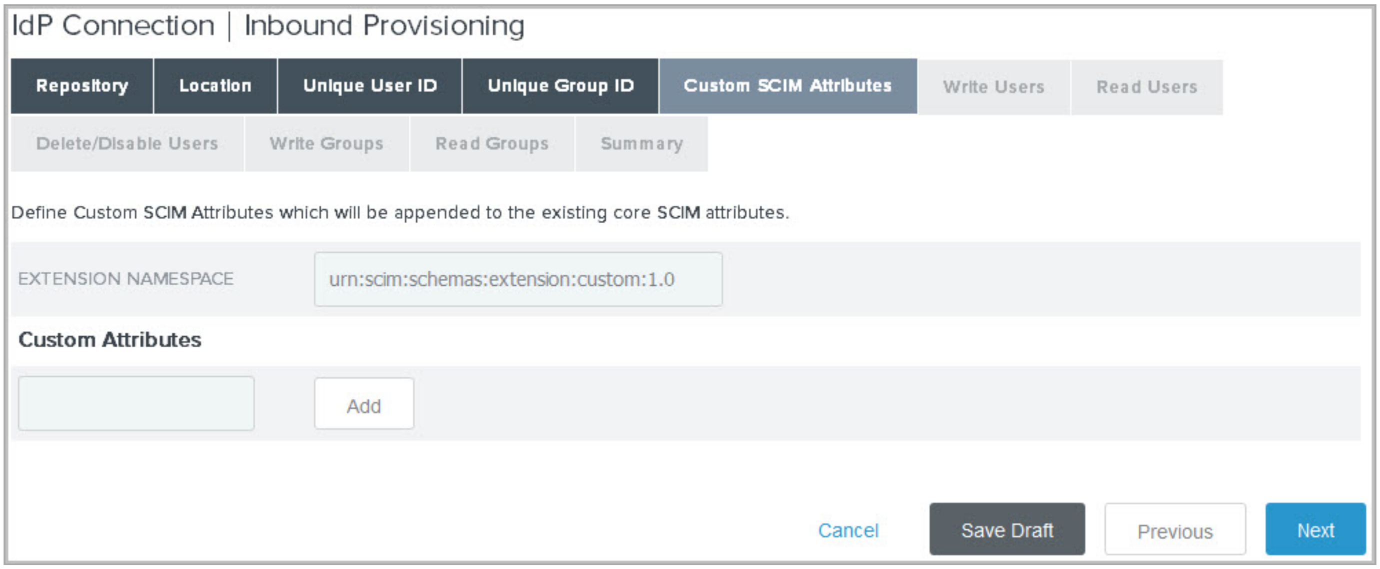
```
{
  "userName":"CBrown",
  "active":true,
  "schemas":[
    "urn:scim:schemas:core:1.0",
    "urn:scim:schemas:extension:custom:1.0"
  ],
  ...
  "urn:scim:schemas:extension:custom:1.0":{
    "supervisor":"JSmith",
    "territories":[
      "Montana",
      "Idaho",
      "Wyoming"
    ],
    "options":{
      "quantity":"10000",
      "strike"  :"5.25",
      "first"   :"2017-12-01",
      "last"    :"2025-03-31"
    },
    "tablets":[
      {
        "model" :"8086",
        "serial":"5500-2020-965",
        "type"  :"office"
      },
      {
        "model" :"8088",
        "serial":"5500-2040-151",
        "type"  :"remote"
      }
    ]
  }
}
```

Attribute Name	Attribute Type	Sub-Attributes (Complex)
`supervisor`	Simple	Not applicable
`territories`	Simple multivalued	Not applicable
`options`	Complex	`quantity`, `strike`, `first`, and `last`
`tablets`	Complex multivalued	`model`, `serial`, and `type`. ⓘ **Note** `type` is a reserved sub-attribute for a complex multivalued attribute.

> **Tip**
>
> For more information about SCIM and attribute types, see the website www.simplecloud.info.

Steps

1. Go to the **Custom SCIM Attributes** tab.



IdP Connection | Inbound Provisioning

Define Custom SCIM Attributes which will be appended to the existing core SCIM attributes.

EXTENSION NAMESPACE: urn:scim:schemas:extension:custom:1.0

Custom Attributes

Choose from:

- To specify a schema extension, enter the URI of the schema extension in the **Extension Namespace** field

> **Tip**
>
> The default value is `urn:scim:schemas:extension:custom:1.0`. You can keep this value if your partner identifies custom attributes by this URI in its SCIM messages.

- To add a custom attribute, enter an attribute name and click **Add**. Repeat this step to add more custom attributes as needed.
- To delete a custom attribute, click **Delete** next to the custom attribute.
- To undo the deletion, click **Undelete**.
- To edit a custom attribute, click **Edit** next to the custom attribute.

  Result:

  The administrative console displays the **Custom SCIM Attribute Options** tab, where you can:

- Change the attribute name.
- Set the attribute as a simple multivalued attribute.
- Add sub-attributes to make it a complex attribute.
- Add sub-attributes and set the attribute as a complex multivalued attribute.

For more information, see Configuring custom SCIM attribute options.

Configuring custom SCIM attribute options

After choosing a custom System for Cross-domain Identity Management (SCIM) attribute on the **Custom SCIM Attribute** tab, you can use the **Custom SCIM Attribute Options** tab to configure the attribute.

About this task

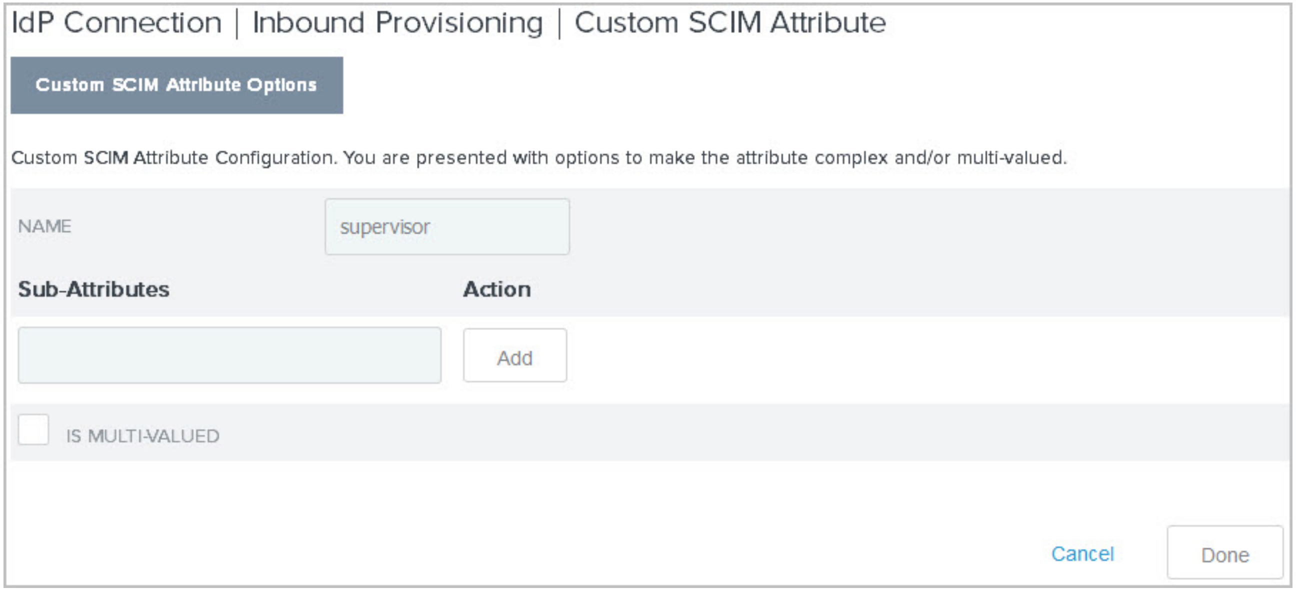
On this tab you can change the attribute name, set the attribute as a simple multivalued attribute, add sub-attributes to make it a complex attribute, and add sub-attributes and set the attribute as a complex multivalued attribute.

Steps

1. Go to the **Custom SCIM Attribute Options** tab.

   Choose from:

   - To change the name of the custom attribute, replace the current value in the **Name** field, and then click **Done**.



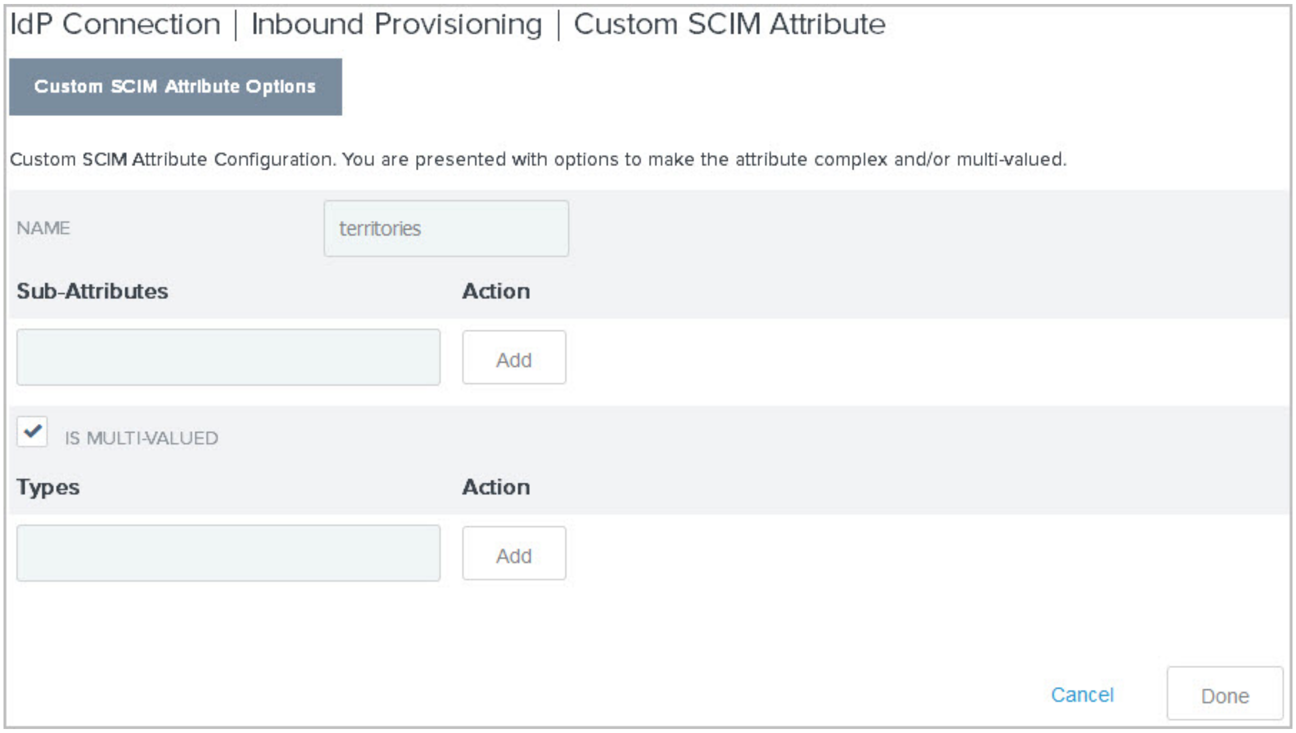
IdP Connection | Inbound Provisioning | Custom SCIM Attribute

Custom SCIM Attribute Options

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: supervisor

- To define the custom attribute as a simple multivalued attribute, select the **Is Multi-valued** check box, and then click **Done**.



IdP Connection | Inbound Provisioning | Custom SCIM Attribute

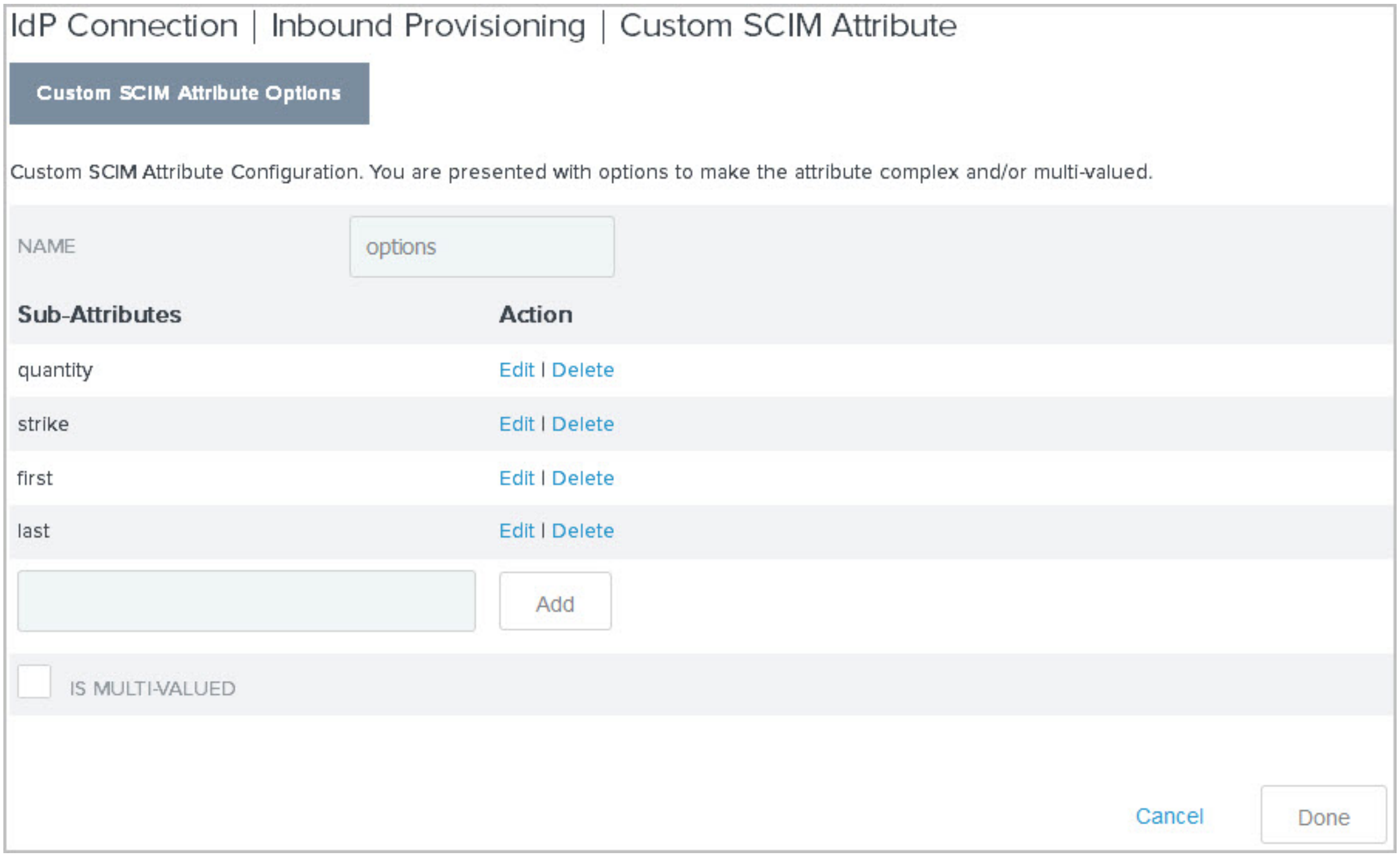
Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: territories

☑ IS MULTI-VALUED

- To define the custom attribute as a complex attribute, enter a sub-attribute, and then click **Add**. Repeat this step to add more sub-attributes as needed, and then click **Done**.

> **Tip**
>
> Click **Edit**, **Update**, or **Cancel** to make or undo a change to the name of a sub-attribute. Click **Delete** or **Undelete** to remove a sub-attribute or cancel the deletion.



IdP Connection | Inbound Provisioning | Custom SCIM Attribute

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: options

Sub-Attributes	Action
quantity	Edit | Delete
strike	Edit | Delete
first	Edit | Delete
last	Edit | Delete

- To define the custom attribute as a complex multivalued attribute, follow these steps.

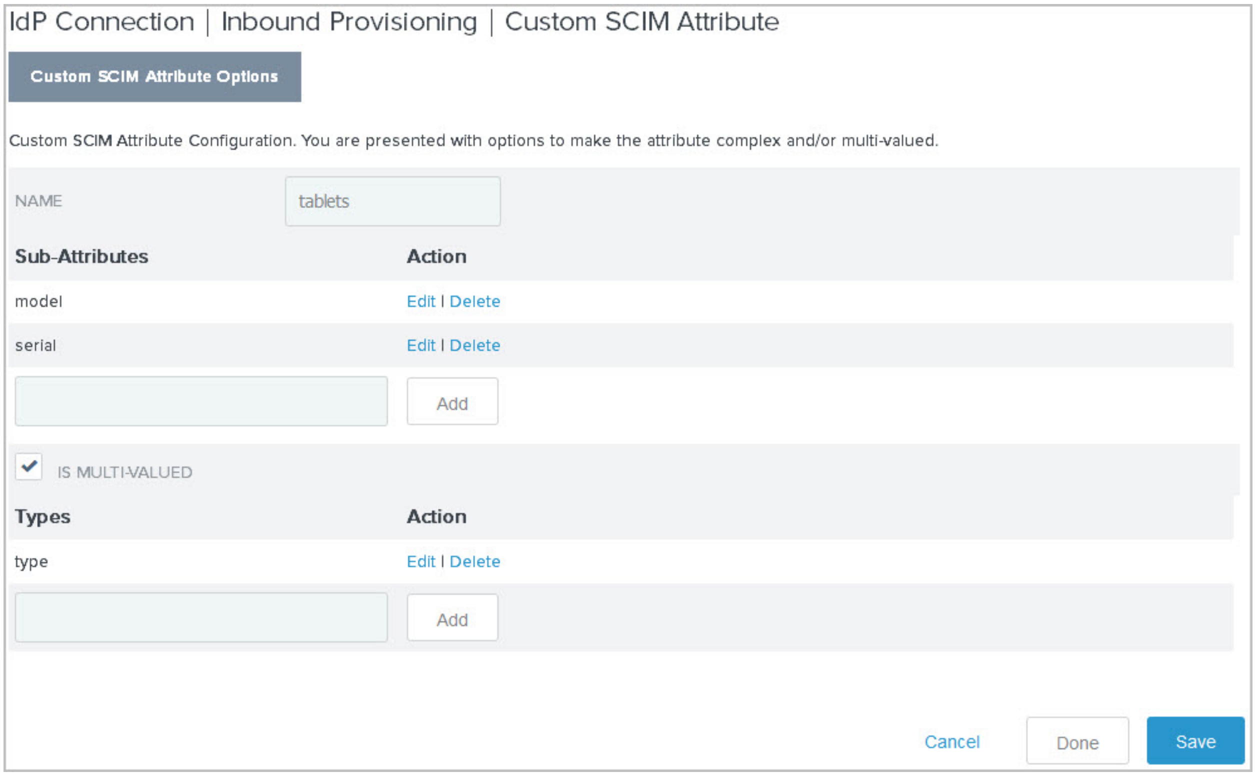
1. Enter a sub-attribute, and then click **Add**. Repeat this step to add more sub-attributes as needed.

   > **Tip**
   >
   > Click **Edit**, **Update**, or **Cancel** to make or undo a change to the name of a sub-attribute. Click **Delete** or **Undelete** to remove a sub-attribute or cancel the deletion.

2. Select the **Is Multi-valued** check box.

3. If you have chosen Active Directory as your user store, you must specify at least one value in the **Types** section for `type`, a reserved sub-attribute for a complex multivalued attribute. For more information, see Specifying the user repository

4. Click **Done**.



IdP Connection | Inbound Provisioning | Custom SCIM Attribute

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: tablets

Sub-Attributes	Action
model	Edit | Delete
serial	Edit | Delete

☑ IS MULTI-VALUED

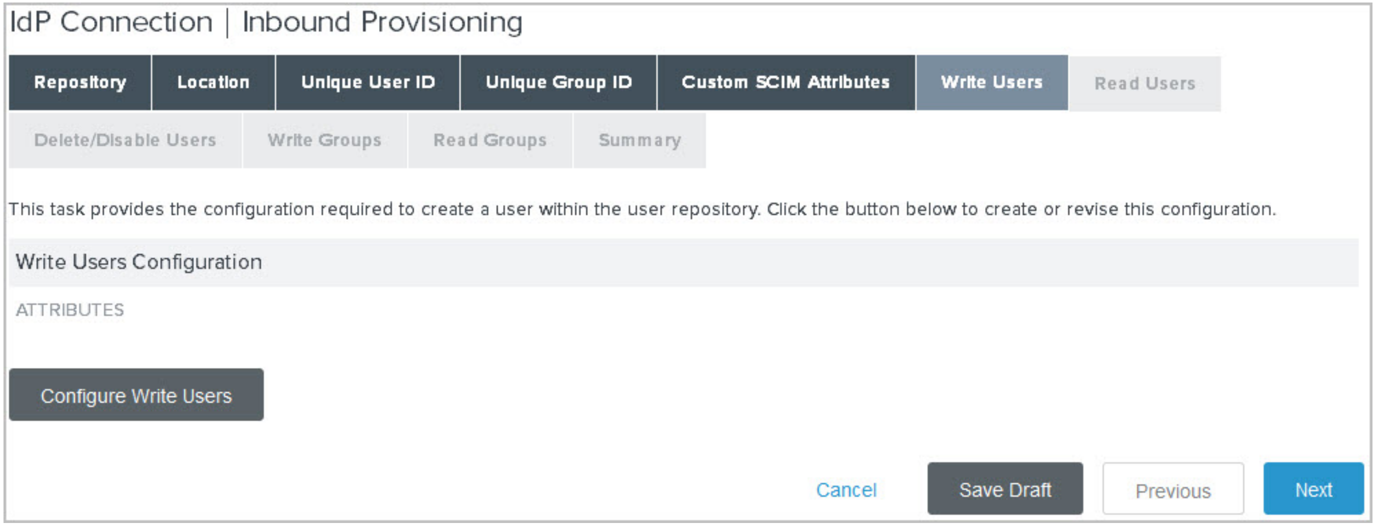
Types	Action
type	Edit | Delete

Writing user information to the datastore

To configure how PingFederate completes create and update operations for user accounts from a System for Cross-domain Identity Management (SCIM) request, identify incoming attributes and map them to datastore attributes.

Steps

1. On the **Write Users** tab, click **Configure Write Users** to continue.



IdP Connection | Inbound Provisioning

This task provides the configuration required to create a user within the user repository. Click the button below to create or revise this configuration.

Write Users Configuration

ATTRIBUTES

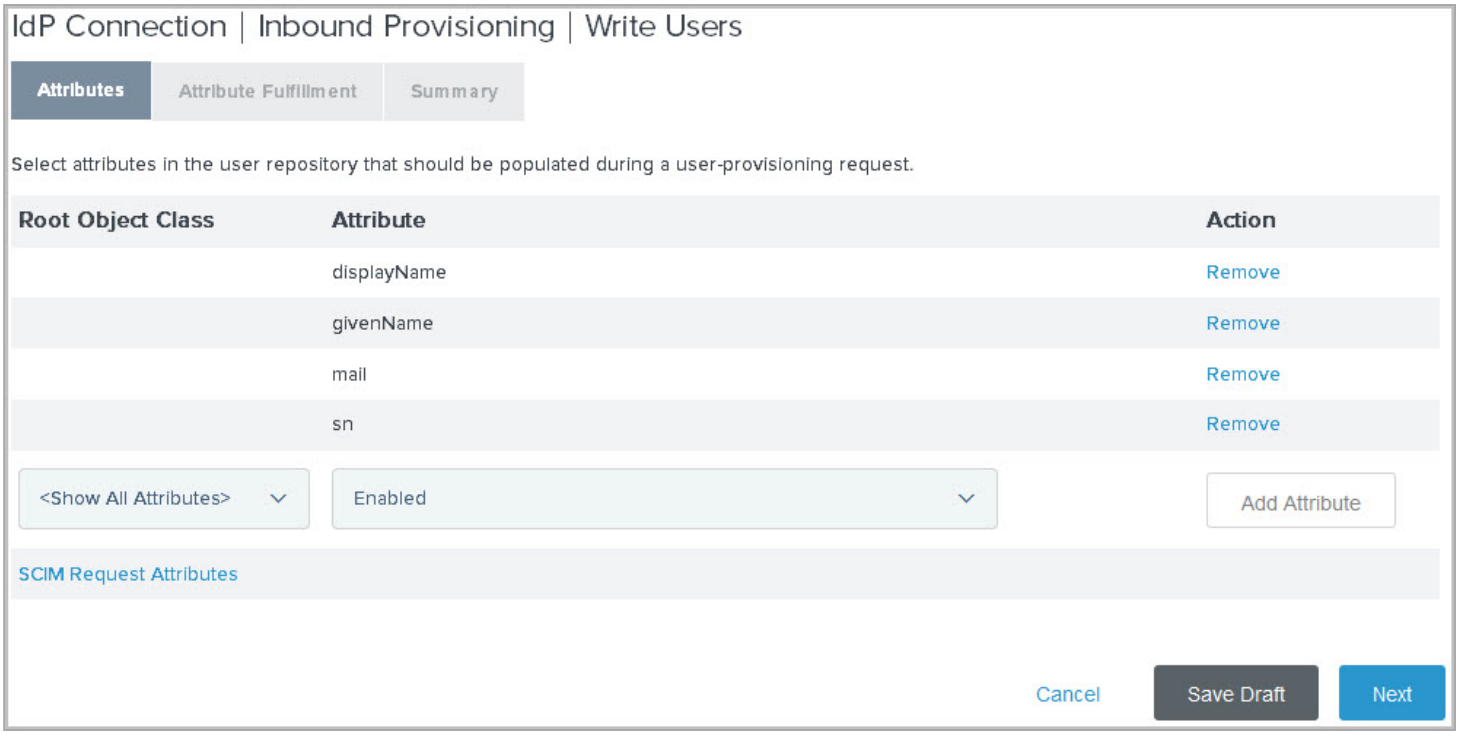
Identifying inbound provisioning attributes for LDAP

When configuring System for Cross-domain Identity Management (SCIM) inbound provisioning, you must identity the attributes you want to provision.

About this task

You can select the datastore attributes you want to provision on the **Attributes** tab.

> **Note**
>
> This tab appears only if you are configuring an LDAP user store for provisioning.



Select attributes in the user repository that should be populated during a user-provisioning request.

Root Object Class	Attribute	Action
	displayName	Remove
	givenName	Remove
	mail	Remove
	sn	Remove
<Show All Attributes>	Enabled	Add Attribute

The following attributes are managed internally by PingFederate and do not require mapping:

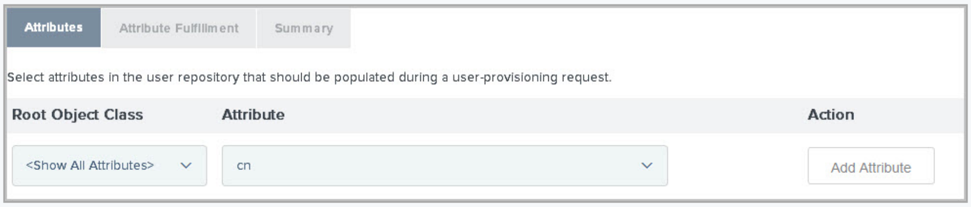
- `objectClass`
- `unicodePwd`
- `objectGUID`
- `userAccountControl`

You can override the internal management of `objectClass` and `unicodePwd` by selecting these attributes and mapping them to SCIM attributes on the **Attribute Fulfillment** tab. In this case, the values you supply are used. The `objectGUID` and `userAccountControl` attributes cannot be overridden and are ignored if selected.

Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.

> **Important**
>
> Do not add `cn` as one of the attributes.



Select attributes in the user repository that should be populated during a user-provisioning request.

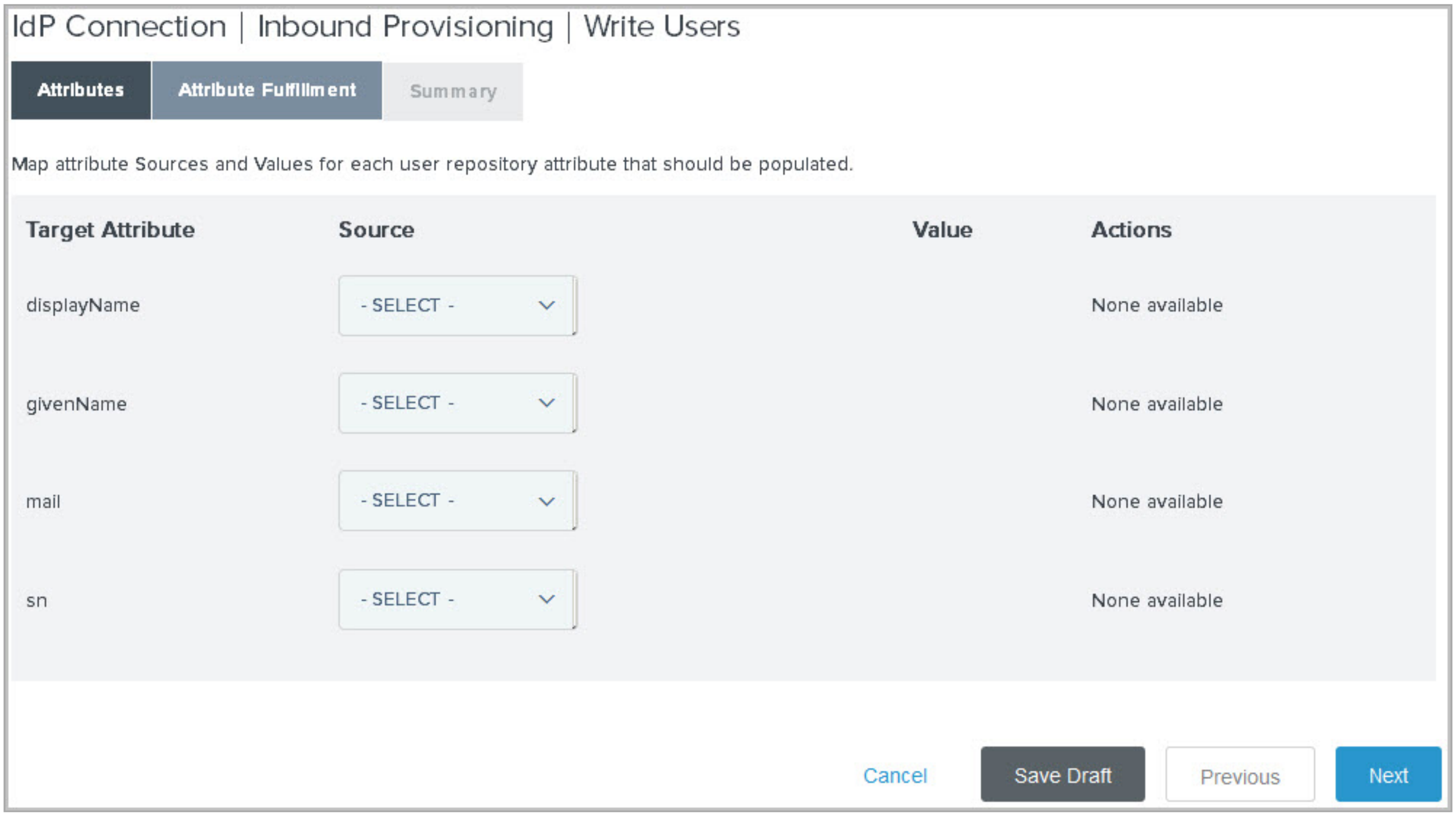
Root Object Class	Attribute	Action
<Show All Attributes>	cn	Add Attribute

2. Repeat the previous step for each attribute requiring provisioning.

Mapping attributes to user accounts

Map attribute values in the System for Cross-domain Identity Management (SCIM) request to user-account attributes.

About this task



Map attribute Sources and Values for each user repository attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available
givenName	- SELECT -		None available
mail	- SELECT -		None available
sn	- SELECT -		None available

Steps

1. On the **Attribute Fulfillment** tab, for each attribute, select a source from the **Source** list and then choose or enter a value. You must map all attributes.

    ◦ `Context`

When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

> **Note**
>
> As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

> **Note**
>
> If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
>
> - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
> - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
>   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
>   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
>   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
> - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

- `Expression`

> **Tip**
>
> Enable OGNL expression by editing the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.ExpressionManager.xml` file. Restart PingFederate after saving the change.
> For a clustered PingFederate environment, edit the `org.sourceid.common.ExpressionManager.xml` file on the console node, sign on to the administrative console to replicate this change to all engine nodes in the **System** > **Server** > **Cluster Management** window, and restart all nodes.

This option provides more complex mapping capabilities, such as transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.

> **Tip**
>
> If two attribute values from a SCIM request need to be mapped to one LDAP attribute value, use an OGNL expression to create it.

- `SCIM User`

When you make this selection, the associated **Value** list populates with defined components of the SCIM request.

- `No Mapping`

Select this option to ignore the **Value** field.

- `Text`

  The value is what you enter. This can be text only, or you can mix text with references to any of the values from the SCIM request using the `${attribute}` syntax.

  > **Tip**
  >
  > You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Done**.

Reviewing user mapping (Write Users) configuration

You can review your user mapping settings before completing configuration.

About this task

The **Summary** tab provides an overview of the inbound provisioning configuration for request mapping.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

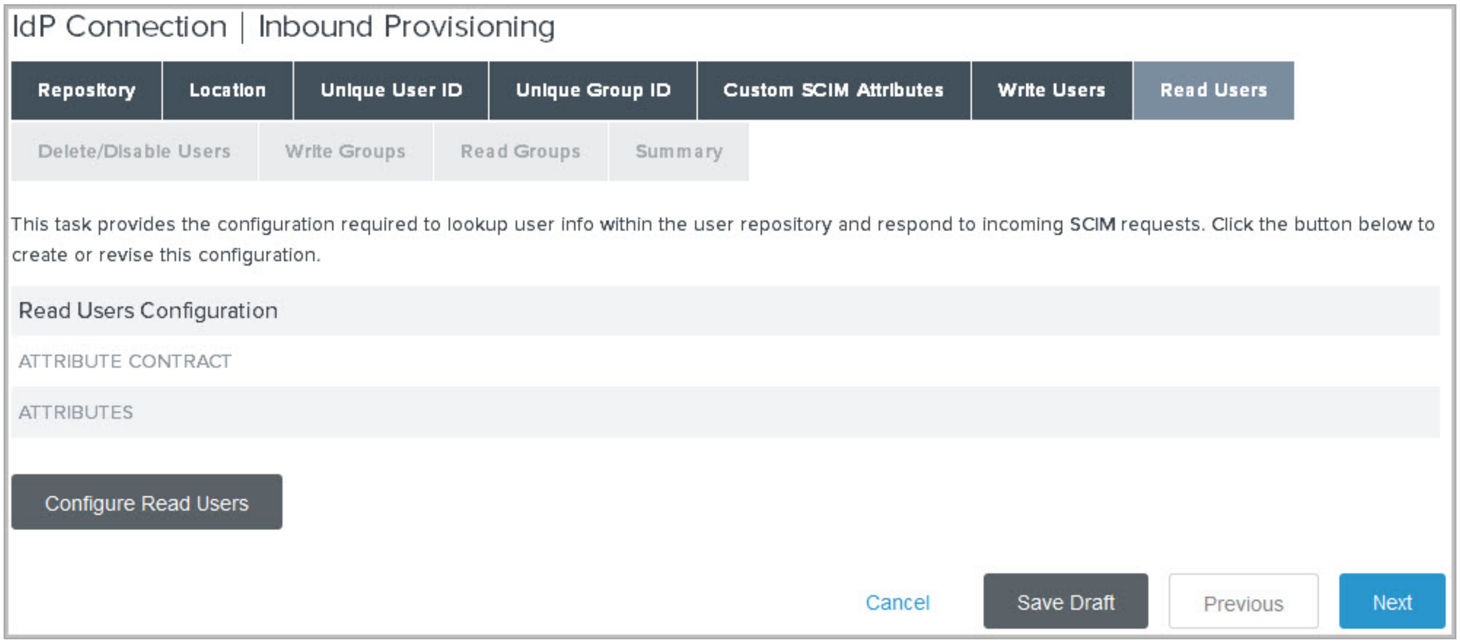
   - To discard your changes, click **Cancel**.

Configuring a SCIM response

To configure a System for Cross-domain Identity Management (SCIM) response to a request to read and return provisioned SCIM attributes, identify and map the user account attributes you want to include.

About this task

Begin this process on the **Read Users** tab.



IdP Connection | Inbound Provisioning

This task provides the configuration required to lookup user info within the user repository and respond to incoming SCIM requests. Click the button below to create or revise this configuration.

Read Users Configuration

ATTRIBUTE CONTRACT

ATTRIBUTES

Steps

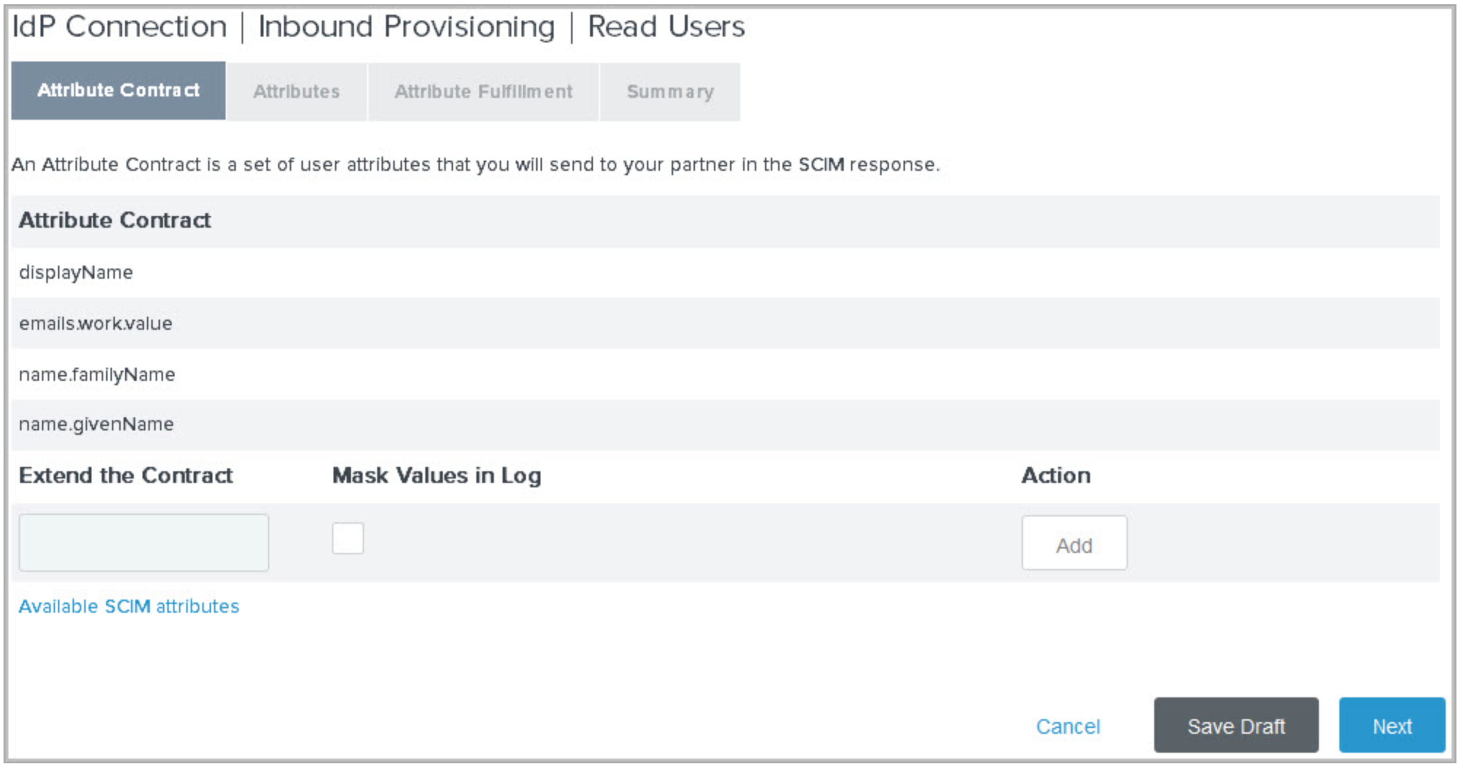
1. On the **Read Users** tab, click **Configure Read Users** to continue.

Identifying expected user attributes for the SCIM response

An attribute contract is a set of user attributes that you and your partner agree will be sent in a System for Cross-domain Identity Management (SCIM) response for this connection.

About this task

On the **Attribute Contract** tab, the attributes you mapped to user account attributes on the **Write Users** tab appear under **Attribute Contract**.



IdP Connection | Inbound Provisioning | Read Users

An Attribute Contract is a set of user attributes that you will send to your partner in the SCIM response.

Attribute Contract
displayName
emails.work.value
name.familyName
name.givenName

Extend the Contract	Mask Values in Log	Action
	□	Add

Optionally, you can mask the values of attributes in the log files that PingFederate writes when it sends the SCIM response.

There are multiple SCIM attributes that are managed internally by PingFederate and are unavailable for inclusion in the attribute contract:

- `id`
- `active`

Steps

1. Click **Available SCIM Attributes** near the lower-left corner of the tab to include additional attributes you want to map in the SCIM response.

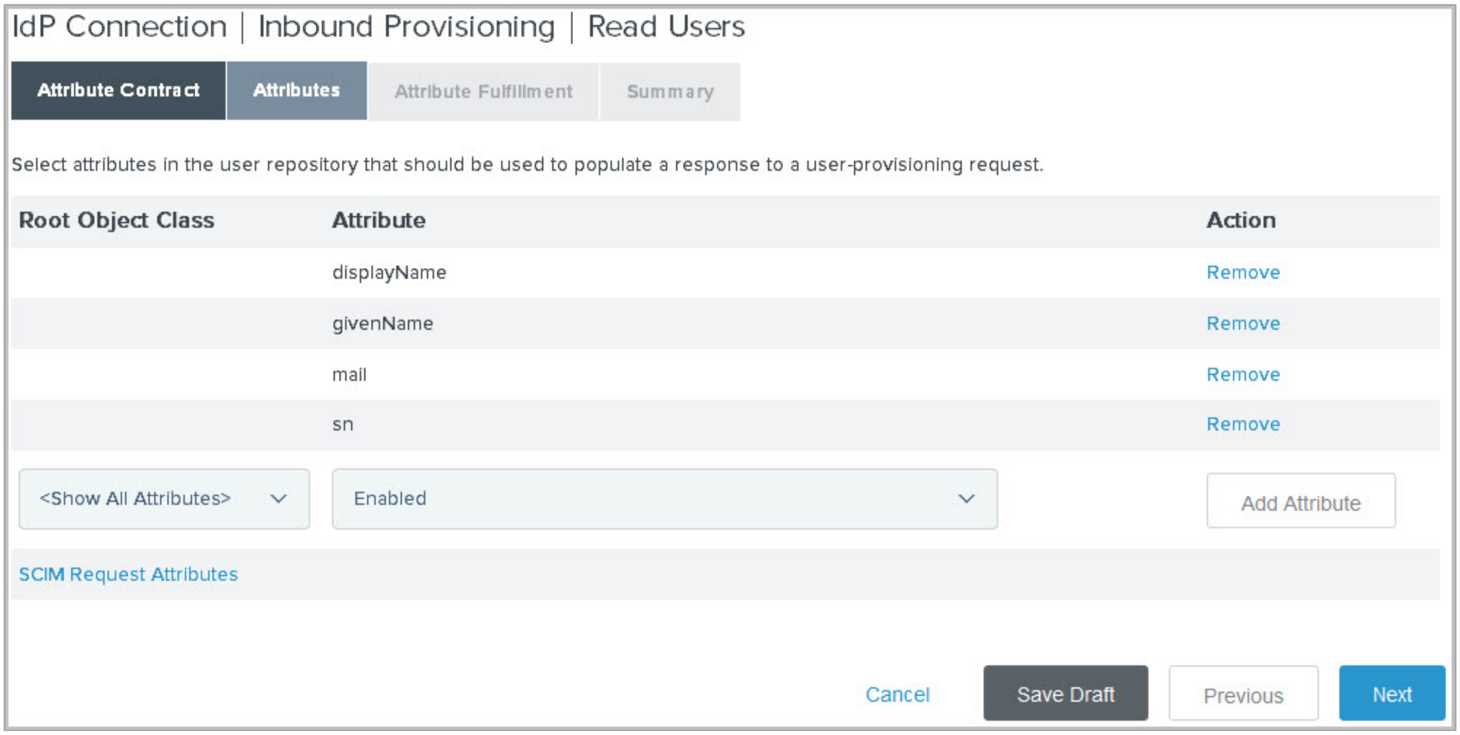
Option	Action
Add an attribute	Enter the attribute name in the text box, select the check box under **Mask Values in Log** as needed, then click **Add**. Attribute names are case-sensitive and must correspond to the attribute names expected by your partner. To see a list of available attributes, click **Available SCIM attributes**.

Option	Action
Modify an attribute name or masking selection	Click **Edit** under **Action** for the attribute, make the change, then click **Update**. + **Note** If you change your mind, make sure you click **Cancel** under **Actions**, not the **Cancel** button, which discards any other changes you might have made in the configuration steps.
Delete an attribute	Click **Delete** under **Action** for the attribute.

Identifying LDAP attributes for the SCIM response

On the **Attributes** tab, you can select the LDAP attributes you want to map to attributes in the System for Cross-domain Identity Management (SCIM) response.

About this task



IdP Connection | Inbound Provisioning | Read Users

Select attributes in the user repository that should be used to populate a response to a user-provisioning request.

Root Object Class	Attribute	Action
	displayName	Remove
	givenName	Remove
	mail	Remove
	sn	Remove

Note

This tab appears only if you are configuring an LDAP user store for provisioning.

Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.
2. Repeat for each attribute requiring provisioning.

Mapping attributes into the SCIM response

Map outgoing user-account attributes to System for Cross-domain Identity Management (SCIM) responses to READ requests.

Steps

1. On the **Attribute Fulfillment** tab, for each target attribute, select a source from the **Source** list, then choose or enter a value. All target attributes must be mapped.

   Choose from:

   - Context

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

     > **Note**
     >
     > Because the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

     > **Note**
     >
     > If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings**window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
     >
     > - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
     > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
     >   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
     >   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
     >   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
     > - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

   - Expression

This option provides more complex mapping capabilities, such as transforming outgoing values into different formats. All of the variables available for text entries are also available for expressions.

> **Tip**
>
> If you need to map an LDAP attribute to two attributes in a SCIM response, use an OGNL expression to create them.

> **Tip**
>
> Enable OGNL expression by editing the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.ExpressionManager.xml` file. Restart PingFederate after saving the change.
> For a clustered PingFederate environment, edit the `org.sourceid.common.ExpressionManager.xml` file on the console node, sign on to the administrative console to replicate this change to all engine nodes in the **System > Server > Cluster Management** window, and restart all nodes.

- **LDAP**

  Values are returned from your query. When you make this selection, the **Value** list populates with the LDAP attributes you identified for this datastore.

- **Identity Store**

  Values are returned from your query. When you make this selection, the **Value** list populates with the Identity Store attributes you identified for this datastore.

- **No Mapping**

  Select this option to ignore the **Value** field.

- **Text**

  The value is what you enter. This can be text only, or you can mix text with references to any of the values from the SCIM request, using the `${attribute}` syntax.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Done**.

Reviewing SCIM response (Read Users) configuration

You can review your System for Cross-domain Identity Management (SCIM) response mapping before completing configuration.

About this task

The **Summary** tab provides an overview of the inbound provisioning configuration for SCIM response mapping.

Steps

1.
    - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
    - To keep your changes, click **Done** and continue with the rest of the configuration.

      > **Tip**
      >
      > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

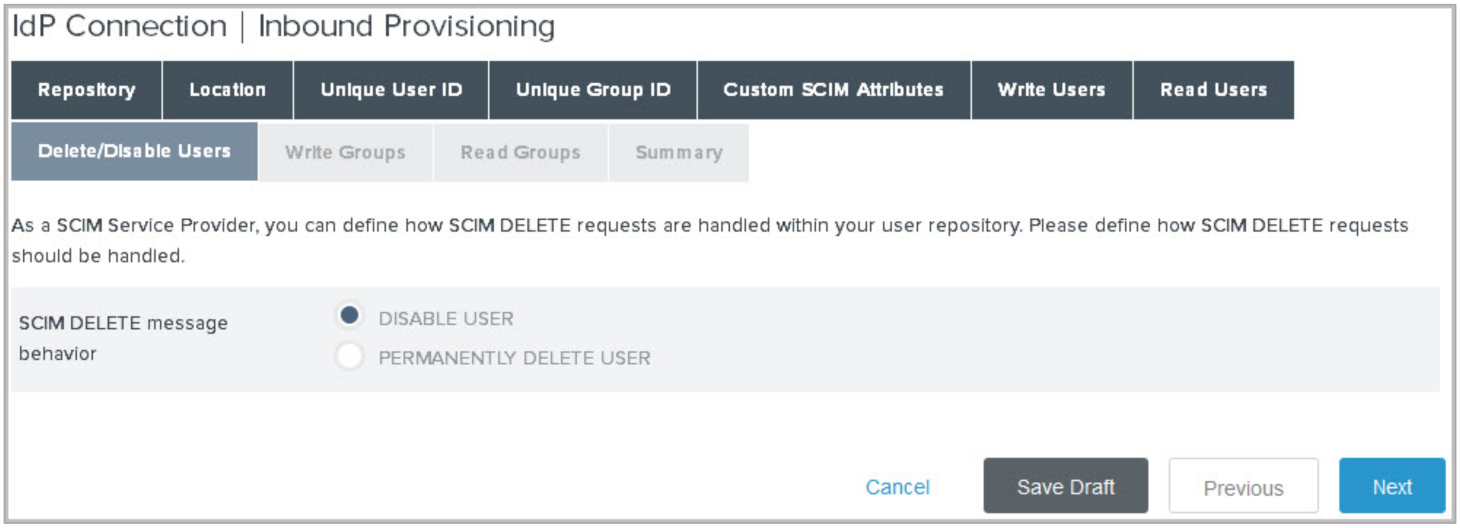
    - To discard your changes, click **Cancel**.

Configuring the handling of SCIM delete requests

You can use the **Delete/Disable Users** tab to define how System for Cross-domain Identity Management (SCIM) delete requests are handled within your user datastore.

About this task

> **Important**
>
> If the group support option is enabled, when PingFederate receives a SCIM delete request for a group, it always removes the specified group from the datastore.



As a SCIM Service Provider, you can define how SCIM DELETE requests are handled within your user repository. Please define how SCIM DELETE requests should be handled.

SCIM DELETE message behavior
● DISABLE USER
○ PERMANENTLY DELETE USER

> **Note**
>
> This tab appears only if you are configuring an LDAP user store for provisioning.

Steps

1. Click one of the two available options for **SCIM DELETE message behavior**.

Choose from:

- Click **Disable User** to make the user inactive within the datastore. This approach is preferred in situations where accounts must be retained for auditing reasons.

  To be SCIM compliant when deleting users, PingFederate returns an HTTP 404 response code for all subsequent operations related to the user-effectively treating the user as if they have been deleted from the LDAP user store. For more information, see SCIM specifications.

> **Caution**
>
> If the user is disabled through another method, PingFederate still treats that user as if they have been deleted and returns HTTP 404 response codes for all subsequent requests.

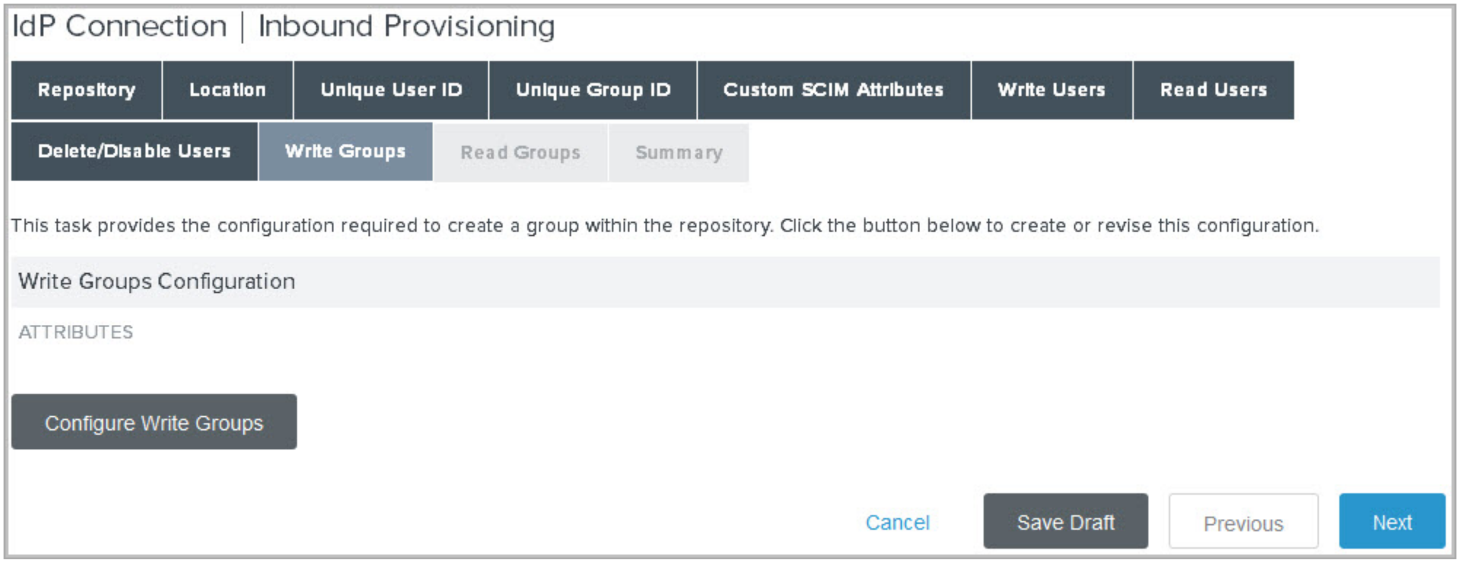
- Click **Permanently Delete User** to remove the user from the datastore.

Writing group information to the datastore

To configure how PingFederate completes create and update operations for groups from a System for Cross-domain Identity Management (SCIM) request, identify incoming attributes and map them to datastore attributes.

About this task

You can identify and map these attributes on the **Write Groups** tab.



This task provides the configuration required to create a group within the repository. Click the button below to create or revise this configuration.

Write Groups Configuration

ATTRIBUTES

Steps

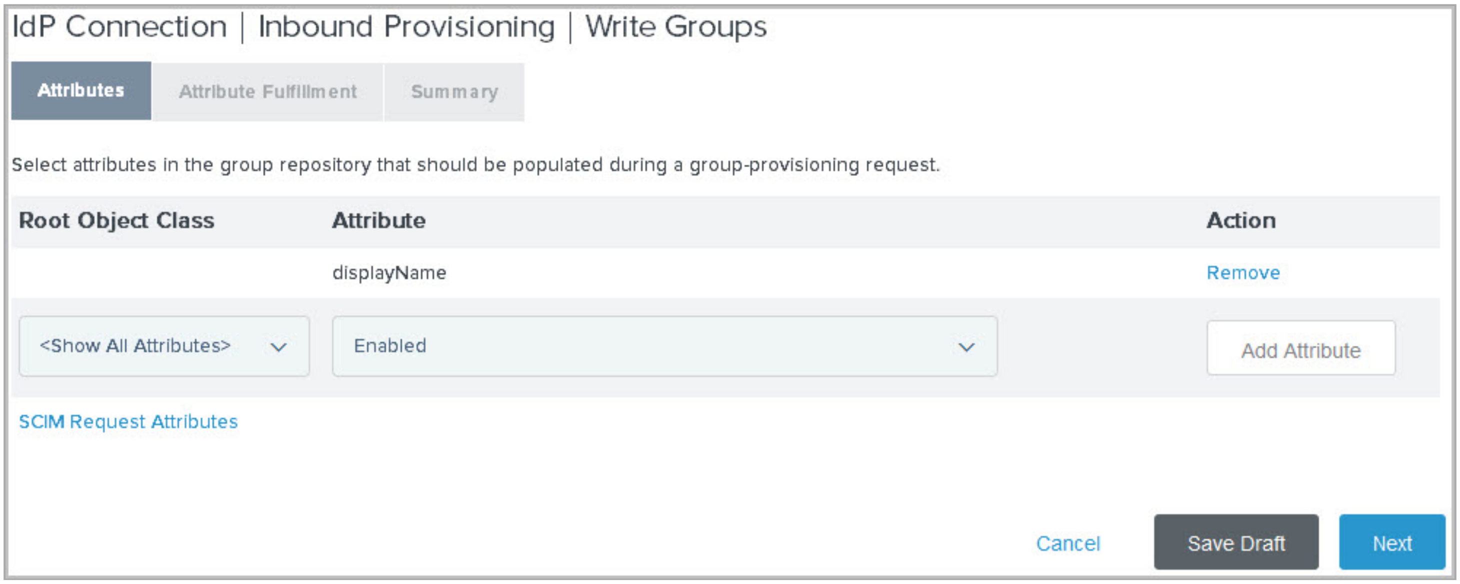
1. Click **Configure Write Groups** to continue.

Identifying inbound provisioning group attributes for LDAP

You must identify the datastore attributes you want to provision when writing group information to the datastore.

About this task

You can identify these attributes on the **Attributes** tab.



IdP Connection | Inbound Provisioning | Write Groups

Select attributes in the group repository that should be populated during a group-provisioning request.

Root Object Class	Attribute	Action
	displayName	Remove
<Show All Attributes>	Enabled	Add Attribute

> **Note**
>
> This tab only appears if you are configuring an LDAP user store for provisioning and the **User and Group Support** option is selected on the **Connection Type** tab.

PingFederate internally manages several attributes that do not require mapping:

- `objectClass`
- `objectGUID`
- `member`

You can override the internal management of `objectClass` by selecting and mapping it to a System for Cross-domain Identity Management (SCIM) attribute on the **Attribute Fulfillment** tab. In this case, the values you supply are used. The `objectGUID` and member attributes cannot be overridden and are ignored if selected.

Steps

1. Select a root object class and an attribute from the lists, and then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Mapping attributes to groups

Map attribute values in the System for Cross-domain Identity Management (SCIM) request to group attributes.

About this task

Steps

1. On the **Attribute Fulfillment** tab, for each attribute, select a source from the list and then choose or enter a value. You must map all target attributes.

   - `Context`

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

     > **Note**
     >
     > As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

     > **Note**
     >
     > If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
     >
     > - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
     > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
     >   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
     >   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
     >   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
     > - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field.
     >   This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

   - `Expression`

     > **Tip**
     >
     > Enable OGNL expression by editing the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.ExpressionManager.xml` file. Restart PingFederate after saving the change.
     > For a clustered PingFederate environment, edit the `org.sourceid.common.ExpressionManager.xml` file on the console node, sign on to the administrative console to replicate this change to all engine nodes in the **System > Server > Cluster Management** window, and restart all nodes.

     This option provides more complex mapping capabilities, transforming outgoing values into different formats. All of the variables available for text entries are also available for expressions.

> **Tip**
>
> If an LDAP attribute needs to be mapped to two attributes in a SCIM response, use an OGNL expression to create them.

- `LDAP`

  Values are returned from your query. When you make this selection, the **Value** list populates with the LDAP attributes you identified for this datastore.

- `Identity Store`

  Values are returned from your query. When you make this selection, the **Value** list populates with the Identity Store attributes you identified for this datastore.

- `No Mapping`

  Select this option to ignore the **Value** field.

- `Text`

  The value is what you enter. This can be text only, or you can mix text with references to any of the values from the SCIM request, using the `${attribute}` syntax.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}` . The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Done**.

Reviewing group mapping (Write Groups) configuration

You can review your group mapping settings before completing configuration.

About this task

The **Summary** tab provides an overview of the inbound provisioning configuration for request mapping.

Steps

1. - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

   > **Tip**
   >
   > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

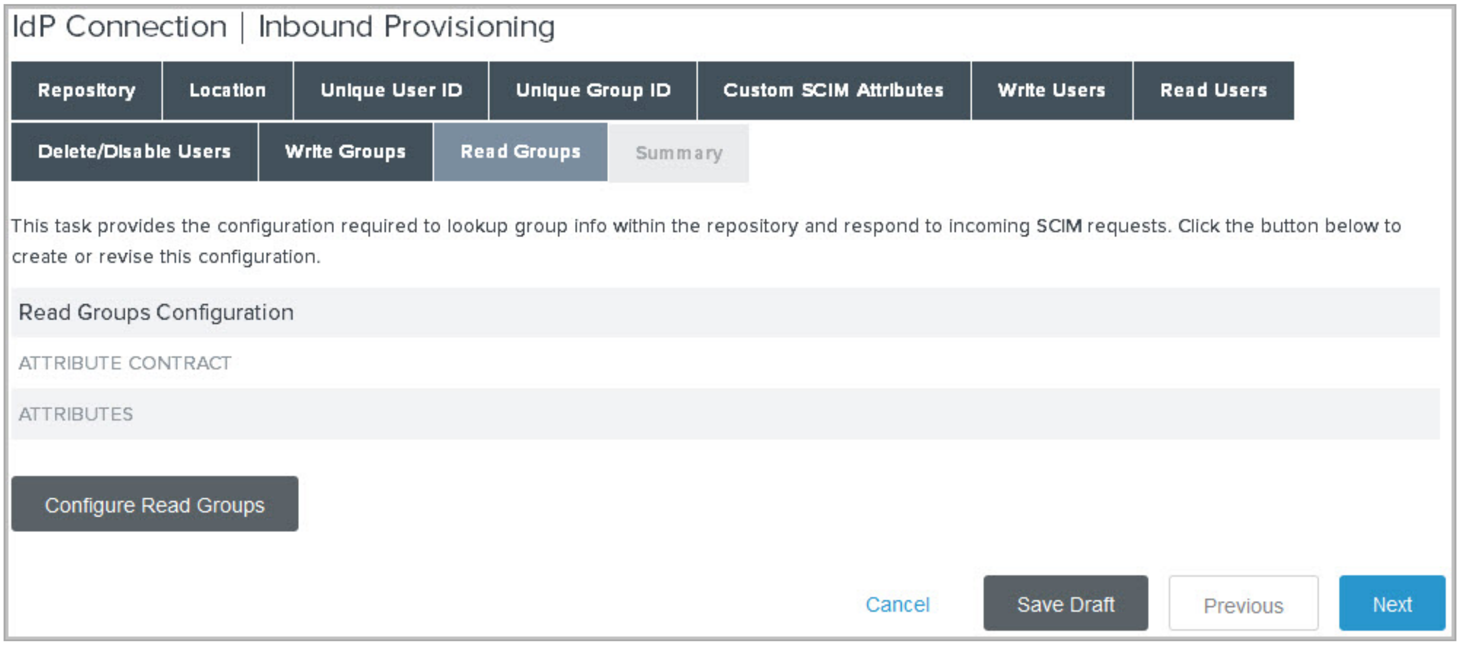
- ◦ To discard your changes, click **Cancel**.

Configuring a SCIM response for groups

To configure a System for Cross-domain Identity Management (SCIM) response to a request to read and return provisioned SCIM attributes, identify and map the group attributes you want to include.

About this task

You can begin to identify and map these attributes on the **Read Groups** tab.



This task provides the configuration required to lookup group info within the repository and respond to incoming SCIM requests. Click the button below to create or revise this configuration.

Steps

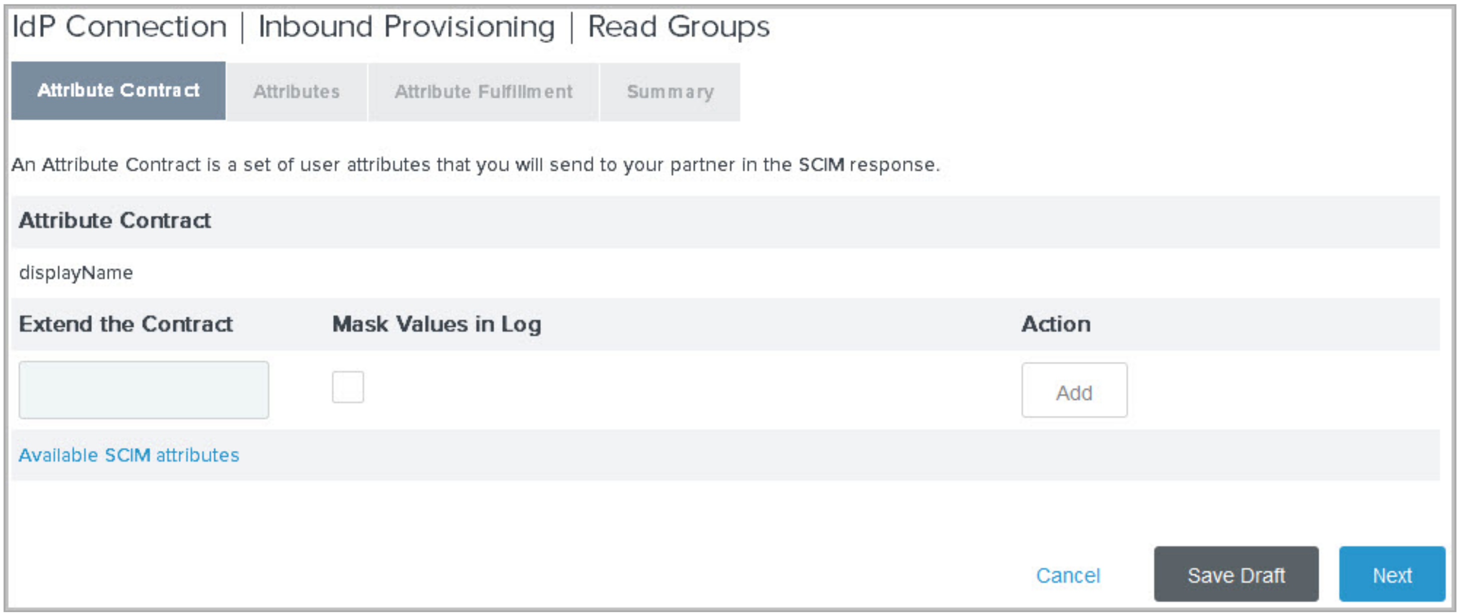
- Click **Configure Read Groups** to continue.

Identifying expected group attributes for the SCIM response

On the **Attribute Contract** tab, you can identify the group attributes for you and your partner.

About this task

An attribute contract is a set of group attributes that you and your partner have agreed will be sent in a System for Cross-domain Identity Management (SCIM) response for this connection. The attributes you mapped to group attributes on the **Write Groups**tab appear at the top of the tab.



An Attribute Contract is a set of user attributes that you will send to your partner in the SCIM response.

Attribute Contract		
displayName		
Extend the Contract	**Mask Values in Log**	**Action**
	□	Add
Available SCIM attributes		

Click **Available SCIM Attributes** near the lower-left corner of the tab to include additional attributes you want to map in the SCIM response.

Optionally, you can mask the values of attributes in the log files that PingFederate writes when it sends the SCIM response.

There are several SCIM attributes that are managed internally by PingFederate and are unavailable for inclusion in the attribute contract:

- `id`
- `members`

Steps

1. To add an attribute, enter the attribute name in the text box, select the check box under **Mask Values in Log** as needed, and click **Add**.

> **Note**
>
> Attribute names are case-sensitive and must correspond to the attribute names expected by your partner. To see a list of available attributes, click **Available SCIM attributes**.

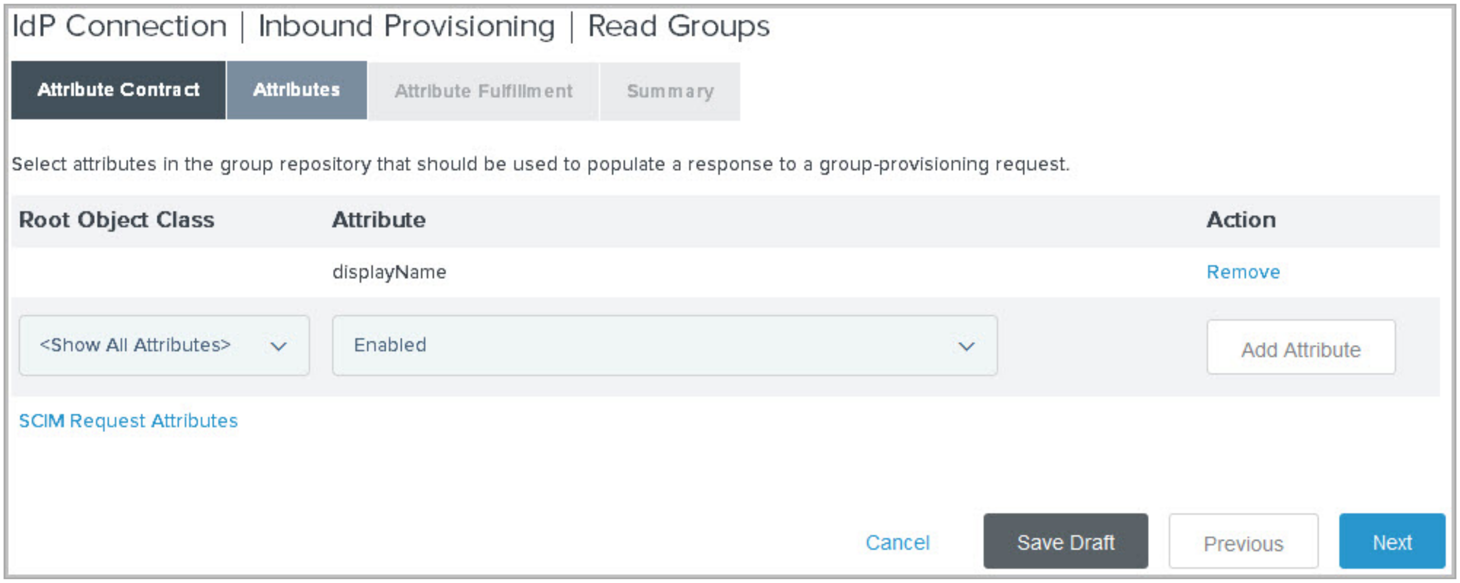
Choice	Action
Modify an attribute name or masking selection	Click **Edit** under **Action** for the attribute, make the change, then click **Update**. **Note** If you change your mind, ensure that you click **Cancel** under **Actions** and not the **Cancel** button, which discards any other changes you might have made in the configuration steps.

Choice	Action
Delete an attribute	Click **Delete** under **Action** for the attribute.

Identifying LDAP group attributes for the SCIM response

On the **Attributes** tab, select the LDAP attributes you want to map to attributes in the System for Cross-domain Identity Management (SCIM) response.

About this task



IdP Connection | Inbound Provisioning | Read Groups

Select attributes in the group repository that should be used to populate a response to a group-provisioning request.

Root Object Class	Attribute	Action
	displayName	Remove
<Show All Attributes>	Enabled	Add Attribute

> **Note**
>
> This tab only appears if you are configuring an LDAP user store for provisioning and the **User and Group Support** option is selected on the **Connection Type** tab.

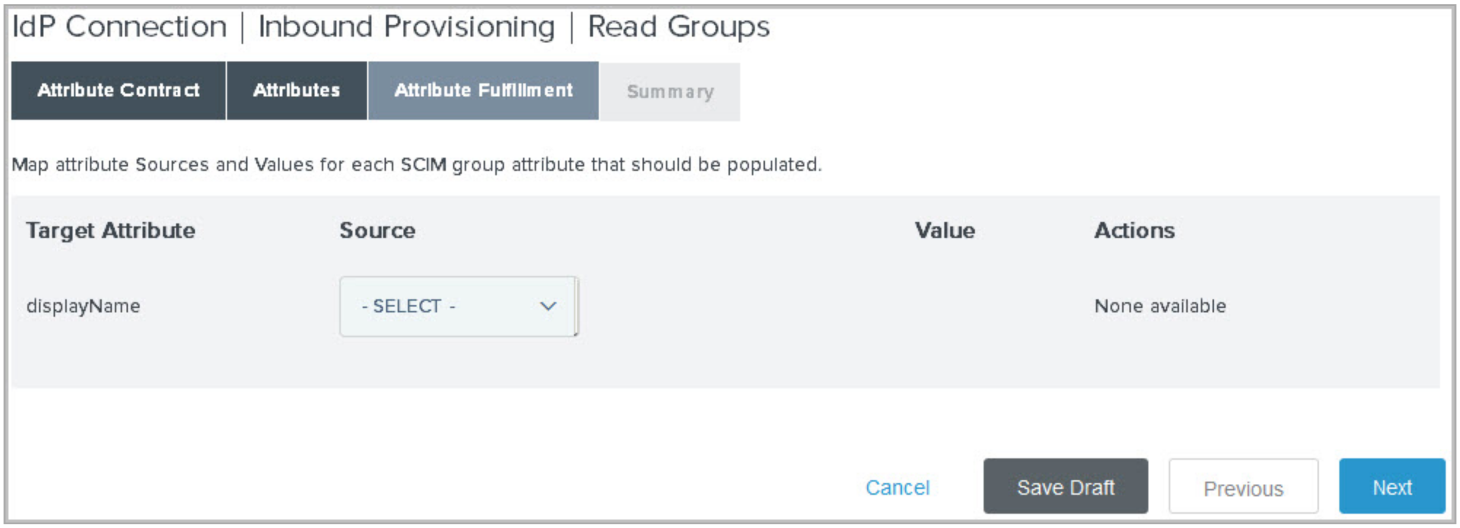
Steps

1. Select a root object class and an attribute from the lists, and then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Mapping group attributes into SCIM response

Map outgoing group attributes to System for Cross-domain Identity Management (SCIM) responses to READ requests.

About this task



IdP Connection | Inbound Provisioning | Read Groups

Map attribute Sources and Values for each SCIM group attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available

Steps

1. On the **Attribute Fulfillment** tab, for each attribute, select a source from the **Source** list and then choose or enter a value. You must map all attributes.

   - `Context`

     When selected, the **Value** list populates with the available context of the transaction. Select the desired context from the list.

     > **Note**
     >
     > As the `HTTP Request` context value is retrieved as a Java object rather than text, use OGNL expressions to evaluate and return values.

     > **Note**
     >
     > If you are configuring an **OAuth Attribute Mapping** configuration and have added `PERSISTENT_GRANT_LIFETIME` as an extended attribute in the **Authorization Server Settings** window, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions or the per-client `Persistent Grants Max Lifetime` setting.
     >
     > - To set lifetime based on the per-client `Persistent Grants Max Lifetime` setting, select **Context** from the **Source** list and **Default Persistent Grant Lifetime** from the **Value** list.
     > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression in the **Value** field.
     >   If the expression returns a positive integer, the value represents the lifetime of the persistent grant in minutes.
     >   If the expression returns the integer 0, PingFederate does not store the grant and does not issue a refresh token.
     >   If the expression returns any other value, PingFederate sets the lifetime of the persistent grant based on the per-client **Persistent Grants Max Lifetime** setting.
     > - To set a static lifetime, select **Text** from the **Source** list and enter a static value in the **Value** field. This is suitable for testing purposes, or cases where the persistent grant lifetime must always be set to a specific value.

   - `Expression`

> **Tip**
>
> Enable OGNL expression by editing the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.common.ExpressionManager.xml` file. Restart PingFederate after saving the change.
> For a clustered PingFederate environment, edit the `org.sourceid.common.ExpressionManager.xml` file on the console node, sign on to the administrative console to replicate this change to all engine nodes in the **System** > **Server** > **Cluster Management** window, and restart all nodes.

This option provides more complex mapping capabilities, transforming outgoing values into different formats. All of the variables available for text entries are also available for expressions.

> **Tip**
>
> If an LDAP attribute needs to be mapped to two attributes in a SCIM response, use an OGNL expression to create them.

- `LDAP`

  Values are returned from your query. When you make this selection, the **Value** list populates with the LDAP attributes you identified for this datastore.

- `Identity Store`

  Values are returned from your query. When you make this selection, the **Value** list populates with the Identity Store attributes you identified for this datastore.

- `No Mapping`

  Select this option to ignore the **Value** field.

- `Text`

  The value is what you enter. This can be text only, or you can mix text with references to any of the values from the SCIM request, using the `${attribute}` syntax.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

2. Click **Done**.

Reviewing SCIM response for groups (Read Groups) configuration

You can review your System for Cross-domain Identity Management (SCIM) settings before completing configuration.

About this task

The **Summary** tab provides an overview of the inbound provisioning configuration for SCIM response mapping.

Steps

1.
    - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
    - To keep your changes, click **Done** and continue with the rest of the configuration.

      > **Tip**
      >
      > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

    - To discard your changes, click **Cancel**.

Reviewing the inbound provisioning configuration

You can review your inbound provisioning settings before completing the configuration.

Steps

1.
    - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
    - To keep your changes, click **Done** and continue with the rest of the configuration.

      > **Tip**
      >
      > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

    - To discard your changes, click **Cancel**.

Configuring security credentials

You can configure security credentials and requirements as needed.

About this task

The **Credentials** tab provides the launching point for configuring security requirements you might need depending on the federation protocol you are using and the choices you have made.

Steps

- To continue, click **Configure Credentials**.

For more information and configuration steps see:

- Configuring back-channel authentication for outbound messages
- Configuring back-channel authentication for inbound messages

- Configuring digital signatures for identity provider connections
- Managing signature verification settings
- Choosing an encryption certificate (SAML 2.0)
- Choosing a decryption key (SAML 2.0)
- Reviewing IdP credential settings
- Reviewing an IdP connection

IdP connection management

When configuring a profile for the inbound artifact binding, outbound SOAP binding, or provisioning, you must specify back-channel authentication information for sending SOAP messages, artifact resolution requests, and provisioning requests to your partner identity provider (IdP).

Similarly, if you send artifacts, SOAP messages, or provisioning messages to your partner IdP, then you must configure SOAP authentication requirements for receiving SOAP responses, artifact resolution requests, or provisioning requests from your partner.

Back-channel authentication also applies to attribute-request configurations, because this profile always uses the SOAP back channel.

See Configuring back-channel authentication for outbound messages and Configuring back-channel authentication for inbound messages for configuration steps.

Configuring back-channel authentication for outbound messages

You can add and edit configuration settings for back-channel authentication for outbound messages.

Steps

1. On the **Back-Channel Authentication** tab, in the **Send to your partner** section, click **Configure**.
2. On the **Outbound SOAP Authentication Type** tab, choose one or more authentication methods.

   HTTP Basic

   When selected, the administrative console prompts you to enter the credentials on the **Basic SOAP Authentication (Outbound)** tab.You must obtain these credentials from your partner.

   SSL Client Certificate

   Applicable only if you specify an endpoint that uses HTTPS.When selected, the administrative console prompts you to specify your client certificate on the **SSL Authentication Certificate** tab. If you have not yet created or imported the client certificate, click **Manage Certificates** to do so. For more information, see Manage SSL client keys and certificates.

> **Important**
>
> When exporting this client certificate for your partner, choose the **Certificate Only** option.

Digital Signature (Browser SSO profile only)

You select a signing certificate on the **Digital Signature Settings** tab.This option leverages on the digital signature of the message.

Perform validation on partner's SSL server certificate when SSL used

By default, PingFederate validates your partner's HTTPS server certificate, verifying that the certificate chain is rooted by a trusted certificate authority (CA) and that the hostname matches the certificate's common name (CN).Clear the associated check box if you do not want this validation to occur.

These options can be used in any combination or independently.

3. On the **Summary** tab, review your configuration and perform one of the following tasks:

Choose from:

- Amend your configuration by clicking the corresponding tab title, then follow the configuration wizard to complete the task.
- Keep your changes by clicking **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- Discard your changes by clicking **Cancel**.

Configuring back-channel authentication for inbound messages

You can add and edit configuration settings for back-channel authentication for inbound messages.

Steps

1. On the **Back-Channel Authentication** tab, in the **Received from your partner** section, click **Configure**.
2. On the **Inbound Authentication Type** tab, choose one or more authentication methods.

HTTP Basic

When selected, the administrative console prompts you to enter the credentials on the **Basic SOAP Authentication (Inbound)** tab.

> **Important**
>
> If you are configuring more than one connection that uses the artifact or HTTP profile, you must ensure that the username is unique for each connection. You must communicate these credentials to your partner out-of-band.

SSL Client Certificate

When selected, the administrative console prompts you to specify the trust model and the related certificate settings on subsequent windows. See the next step.

Digital Signature (Browser SSO profile only)

You select a signing certificate on the **Signature Verification Settings** tab.This option leverages on the digital signature of the message.

Require SSL

When selected, incoming HTTP transmissions must use a secure channel. This option is selected by default.You can clear the check box if you do not require a secure channel and client certificate authentication.

For SAML 2.0, use these options in any combination or independently. For SAML 1.x, you must enable HTTP Basic authentication, client certificate authentication, or both. You can also add digital signing to ensure message integrity.

3. If you chose **SSL Client Certificate** in the previous step, select a trust model on the **Certificate Verification Method** tab.

Anchored

The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you can also restrict the issuer to a specific Trusted CA to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections. The CA’s certificate must be imported into the PingFederate Trusted CA store on the **Trusted CAs**window..

Unanchored

The partner certificate is self-signed or you want to trust a specified certificate.

> **Note**
>
> When anchored certificates are used between partners, certificates can be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated.
> For more information, see Digital signing policy coordination.

Trust model	Subsequent steps
Anchored	On the **Subject DN** tab: 1. Enter the `Subject D`N of the certificate. 2. Optionally, select the **Restrict Issuer** check box and enter the `Issuer DN` of the certificate. **Important** Consider enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.
Unanchored	On the **SSL Verification Certificate** tab, select the client certification from your partner. If you have not yet imported the client certificate from your partner, click **Manage Certificates** to do so. For more information, see Managing certificates from partners.

4. On the **Summary** tab, review your configuration and perform one of the following tasks:

   Choose from:

   - Amend your configuration by clicking the corresponding tab title, then follow the configuration wizard to complete the task.
   - Keep your changes by clicking **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - Discard your changes by clicking **Cancel**.

Configuring digital signatures for identity provider connections

Managing digital signature settings defines the private key you will use to sign single sign-on (SSO) authentication or attribute requests (optionally) or SAML 2.0 single logout (SLO) messages for this identity provider (IdP).

About this task

This process allows you to include Key Info with the XML message if you and your partner have agreed to this option.

Digital signing applies to service provider (SP)-initiated SSO under SAML 2.0, when specified by your partner agreement, and to either SLO profile using the POST or redirect bindings. Digital signing also applies if you are configuring an Attribute Query profile and have specified that you will sign attribute requests.

The step is not required for SAML 1.x IdP connections.

Steps

1. On the **Digital Signature Settings** tab, select a signing certificate from the **Signing Certificate** list.

If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates**. For more information, see Manage digital signing certificates and decryption keys.

2. **Optional:** Select a **Secondary Signing Certificate** for inclusion in the connection metadata.

> **Note**
>
> You can't add a secondary certificate if the primary certificate has certificate rotation enabled. Also, you can't use a certificate that has rotation enabled as the secondary signing certificate.

3. **Optional:** Select the **Include the certificate in the signature <KeyInfo> element** check box if you have agreed to send your public key with the message.

   Select the **Include the raw key in the signature <KeyValue> element** check box if your partner agreement requires it.

   Select the signing algorithm from the list.

The default selection is **RSA SHA256** or **ECDSA SHA256**, depending on the **Key Algorithm** value of the selected digital signing certificate. Make a different selection if you and your partner have agreed to use a stronger algorithm. For a list of the available signing algorithms and their URIs, see Signing algorithms.

Managing signature verification settings

Under SAML 2.0 specifications, when your site receives any SAML 2.0 messages with the POST or Redirect bindings, the messages must be digitally signed.

About this task

Signing is also always required for the SAML 1.x POST binding and for WS-Federation assertions, as well as incoming SAML 1.1 or 2.0 tokens for WS-Trust STS processing.

Depending on your agreement with this idenity provider (IdP), single sign-on (SSO) assertions, SAML 2.0 artifacts, or SOAP messages might also require signatures.

Steps

1. On the **Signature Verification Settings** tab, click **Manage Signature Verification Settings**.

2. On the **Trust Model** tab, select a trust model on the **Certificate Verification Method** tab.

   Anchored

   The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you can also restrict the issuer to a specific Trusted CA to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections. The CA's certificate must be imported into the PingFederate Trusted CA store in the **Security > Certificate & Key Management > [.wintitle] Trusted CAs**** window.

> **Important**
>
> If you are using the redirect binding for single logout (SLO) or establishing an OAuth assertion grant connection to exchange JSON web tokens (JWTs) for access tokens. you cannot use anchored certificates because SAML 2.0 does not permit certificates to be included using this transport method and the signature verification process for JWTs requires the public keys to validate the digital signatures.

Unanchored

The partner certificate is self-signed or you want to trust a specified certificate.

> **Note**
>
> When anchored certificates are used between partners, certificates can be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated. For more information, see Digital signing policy coordination.

Trust model	Subsequent steps
Anchored	On the **Subject DN** window: 1. Enter the `Subject DN` of the certificate or extract it from your service provider (SP) partner’s certificate if the certificate is stored on an accessible file system. 2. (Optional) Select the **Restrict Issuer** check box and enter the `Issuer DN` of the certificate. Alternatively, extract it from your partner’s certificate. **Important** Consider enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.

Trust model	Subsequent steps
Unanchored	On the **Signature Verification Certificate** window: 1. Select a certificate from the list. If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so. See Managing certificates from partners. 2. (Optional) Select additional certificates. **Note** When configured, PingFederate considers a digital signature valid so long as it can verify the signature using one of the certificates from this list. **Tip** This is useful in situations where your partner has sent you a certificate to replace the current certificate. Adding this second certificate allows PingFederate to continue validating digital signatures as the partner switches to the new signing certificate. It also adds support for the scenario where your partner uses a pool for certificates to sign its messages. Adding these certificates ensures digital signatures can be validated as the partner rotates its signing certificates.

3. On the **Summary** tab, review your configuration and perform one of the following tasks.

Amend your configuration

Click the corresponding tab title and then follow the configuration wizard to complete the task.

Keep your changes

Click **Done** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Choosing an encryption certificate (SAML 2.0)

If `SAML_SUBJECT` is encrypted, either by itself or as part of a whole assertion, then all references to this name identifier in SAML 2.0 single logout (SLO) requests from your site might also be encrypted if the connection uses service provider (SP)-initiated SLO.

About this task

You must also choose a certificate if encryption of the name identifier is required for an Attribute Request profile. For more information, see Specifying XML encryption policy (for SAML 2.0).

Steps

1. **Optional:** Select an option under **Block Encryption Algorithm**.

    > **Important**
    >
    > Because of the import restrictions of some countries, Oracle Server Java SE Runtime Environment (JRE) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, enable the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy. For more information, see the Java 8 release notes in Oracle's documentation.
    > For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide in Oracle's documentation.

    The default selection is **AES-128**.

    For more information about XML block encryption and key transport algorithms, see XML Encryption Syntax and Processing from W3C.

2. Select an option under **Key Transport Algorithm**.

    > **Note**
    >
    > Due to security risks associated with the RSA-v1.5 algorithm used for key transport, it is no longer available for new connections. Existing connections in which this algorithm is configured continue to support it. However, you should upgrade such connections to use a newer algorithm.

    The default selection is **RSA-OAEP**.

3. Select a partner certificate from the list.

    If you have not imported the certificate from your partner, click **Manage Certificates** to do so. For more information see Managing certificates from partners.

Choosing a decryption key (SAML 2.0)

As part of XML encryption, you must identify a certificate and key for PingFederate to use to decrypt incoming assertions or assertion elements.

About this task

For more information on XML encryption, see Specifying XML encryption policy (for SAML 2.0).

Steps

1. Select the primary XML decryption key from the list.

    If you have not created or imported your certificate into PingFederate, click **Manage Certificates**. For more information, see Manage digital signing certificates and decryption keys.

2. **Optional:** Select the secondary XML decryption key from the list.

Reviewing IdP credential settings

You can review chosen credential settings before completing configuration.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click **Done** and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click **Cancel**.

Reviewing an IdP connection

When you finish creating or modifying a connection, you can review the connection settings and toggle the connection status on the **Activation & Summary** tab.

About this task

> **Important**
>
> When creating a new connection, the default connection status is **Enabled** when you reach the **Activation & Summary** tab.
> Regardless of whether you choose to disable a new connection now or later, you must click **Save** on the **Activation & Summary** tab if you want to keep the new connection.

Steps

1.
2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
   - To keep your changes, click **Save**.
   - To discard your changes, click **Cancel**.

Result

The **SSO Application Endpoint** provides a sample URL at the `/sp/startSSO.ping` application endpoint that webmasters or web application developers at your site might use to invoke single sign-on (SSO) for the connection. For a list of supported parameters, see Viewing SP application endpoints.

If you have selected the **No Mapping** option on the **Identity Mapping** tab, the **Summary & Activation** tab does not show the **SSO Application Endpoint** sample URL.

OpenID Connect Relying Party support

PingFederate can leverage identities from OpenID Providers (OPs) to complete browser single sign-on (SSO) requests.

In this use case, PingFederate is an OAuth client, specifically a relying party (RP) to the OP. PingFederate supports both the Basic Client and the Implicit Client profiles.

The setup involves establishing an IdP connection to the OP. PingFederate retrieves identity information from the OP and passes the end-user claims, which are user attributes in an ID token, to one or more target applications. This configuration allows administrators to take advantage of their existing last-mile integration and expand the horizon of their applications to additional partners using the OpenID Connect protocol.

If the OP supports the OpenID Connect Discovery specification, the connection setup is expedited by loading the metadata from the OP. Based on the discovery specification, PingFederate makes a direct HTTP GET request to the `/.well-known/openid-configuration` endpoint at the OP and populates the attribute contract and the protocol settings of the connection automatically. Manual adjustments can be made during the connection setup or at a later time.

Additionally, you can refresh the protocol settings by reloading the metadata from the OP at any time. If additional claims are supported, PingFederate adds them to the attribute contract, so that they can be mapped to the target applications later. In the event that the previously supported claims have been dropped by the OP from the metadata, they remain in the attribute contract. While the existing mapping configuration might not be adversely affected, runtime errors might occur if certain attributes are no longer available to the target applications. If runtime errors occur, review the server log and modify the mapping configuration accordingly.

If applicable, administrators can define additional request parameters, which can be included in the authentication requests to support OP-specific use cases. Administrators can restrict the values to those defined in the configuration. Alternatively, administrators can allow the target applications to optionally override the values at runtime. As an added security measure, administrators can protect the requested authentication context, `acr_values`, and authentication requirement, `prompt`, so that these parameters in the authentication requests cannot be overridden by the target applications. By default, PingFederate sends these request parameters through multiple query parameters unsigned. Optionally, administrators can configure PingFederate to send request parameters through a request object by value, in which case request parameters are represented by individual claims in a signed JSON Web Token (JWT). When the OP receives the authentication request, it can validate the integrity of the request parameters based on the digital signature in the JWT. For more information, see the section explaining passing a request object by value in the OpenID Connect specification at openid.net/specs/openid-connect-core-1_0.html#RequestObject.

Processing steps

1. A user starts a browser SSO request at the `/sp/startSSO.ping` service provider (SP) application endpoint or the `/sp/init_login.ping` SP protocol endpoint.
2. The relying party, PingFederate, sends to the OP an authentication request though the browser containing the desired request parameters, such as `response_type`, `scope` and `redirect_uri`, with or without any custom query parameters.
3. The OP prompts the user to authenticate and authorize, as needed.
4. The OP sends the user back to PingFederate, at the `redirect_uri` parameter value, through the browser with an authorization code for the Basic Client profile, or an ID token and an access token if specified in the configuration for the Implicit Client profile.
5. Applicable only to the Basic Client profile, PingFederate sends a token request with the authorization code to the OP at its token endpoint, directly through a back-channel HTTP POST request. The OP returns to PingFederate an access token and an ID token.

6. PingFederate validates the ID token.
7. PingFederate passes the end-user claims to the target application through an SP adapter instance or an authentication policy contract through the browser.

   The access token, if any, can also be passed to the target application, so that API security use cases can be layered on top of the browser-based SSO request.

> **Note**
>
> If the UserInfo endpoint is included as part of the connection settings and an access token is provided by the OP, PingFederate also retrieves claims from the OP before the last step.

For more information about OpenID Connect, see openid.net/connect.

Creating an OpenID Connect IdP connection

In the **IdP Connections** window, create an OpenID Connect (OIDC) identity provider (IdP) connection to take advantage of your existing last-mile integration and expand the horizon of your applications to additional partners using the OpenID Connect protocol.

Steps

1. Go to **Authentication** > **Integration** > **IdP Connections**, and then create a new IdP connection.
2. On the **Connection Type** tab, select the **Browser SSO Profiles** check box, and in the **Protocol** list, select **OpenID Connect**. Click **Next.**

   > **Note**
   >
   > When OpenID Connect is the chosen protocol, the other types become unavailable.

3. On the **Connection Options** tab, you can enable just-in-time (JIT) provisioning, OAuth attribute mapping, which requires the OAuth 2.0 authorization server role, or both. Click **Next**.

   > **Note**
   >
   > For simplicity, this topic focuses on managing OpenID Connect IdP connection settings.

4. On the **General Info** tab:
   1. Provide the required information, including:

      `Issuer`

      The Issuer Identifier of the OpenID Provider (OP).

      `Connection Name`

      A plain-language identifier for the connection; for example, a company or department name. This name is displayed in the connection list on the administrative console.

`Client ID`

The client ID to communicate with the OP.

This client represents PingFederate and is created and managed at the OP. For more information, see the documentation provided by the OP.

`Client Secret`

The client secret to communicate with the OP.Applicable only when the client representing PingFederate supports the Basic Client profile. For more information, see step 10.

2. **Optional:** Click **Load Metadata**.

> **Tip**
>
> Loading metadata from the OpenID Provider (OP) expedites the connection setup. You can also update an existing connection by reloading metadata.

5. On the **Browser SSO** tab, click **Configure Browser SSO**.

6. On the **User-Session Creation** tab, click **Configure User-Session Creation**.

7. On the **Identity Mapping** tab, you have three choices:

   Choose from:

   - Select the **No Mapping** check box if you plan on passing end-user claims to the target application through an authentication policy contract in an SP authentication policy.

   - Select the **Account Mapping** check box if you plan on passing end-user claims to the target application through an SP adapter instance or an authentication policy contract if your PingFederate server is a federation hub that bridges an OP to an SP.

   - Select the **Account Linking** check box if your target application requires account linking.

> **Tip**
>
> End-user claims are basically user attributes found in ID tokens or obtained from the UserInfo endpoint at the OP.

   For illustration, this topic uses the **Account Mapping** configuration.

8. On the **Attribute Contract** tab, extend the attribute contract.

   To mask the attribute values in the log, select the relevant check box for each applicable end-user claim.

> **Note**
>
> If you have chosen to load the metadata from the OP on the **General Info** tab, the attribute contract is populated automatically.

9. On the **Target Session Mapping** tab, click **Map New Adapter Instance** to map end-user claims to the target application through an SP adapter instance or an authentication policy contract.

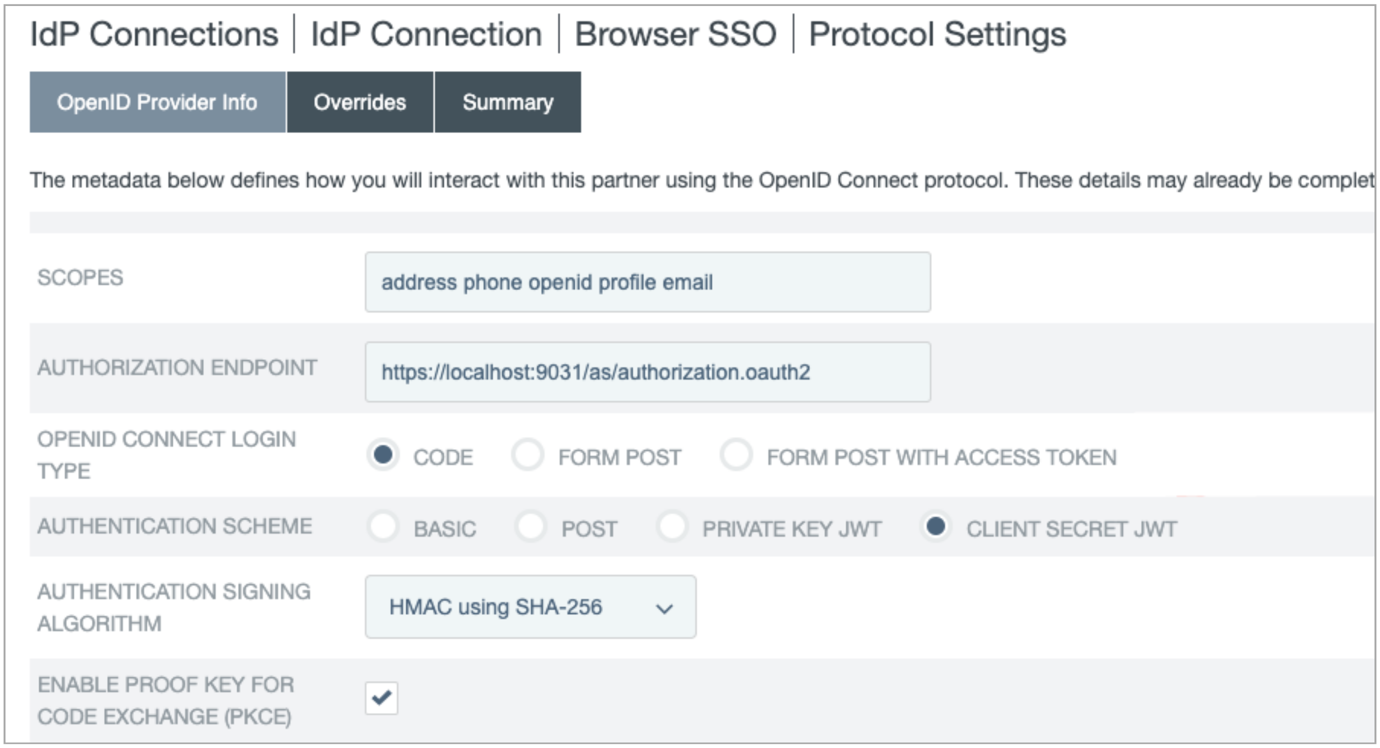
Follow the administrative console to fulfill the SP adapter contract or the authentication policy contract. Like other IdP connections, you can query additional attributes from a datastore, specify issuance criteria, or both. When mapping an attribute, select `Provider Claims` from the **Source** list to map the attribute to an end-user claim.

If your target application requires the associated access token, select `Context` as the source and `Access Token` as the value.

> **Note**
>
> If the client representing PingFederate supports the Basic Client profile, PingFederate always receives an access token from the OP to retrieve an ID token.
> If the client supports the Implicit Client profile, you must select the **Form POST with access token** option in step 10, such that the OP will return an access token and an ID token as part of the authentication and authorization flow.

The **Target Session Mapping** configuration does not apply when the **No Mapping** option is selected on the **Identity Mapping** tab.

10. On the **Protocol Settings** tab, click **Configure Protocol Settings**.
11. On the **OpenID Provider Info** tab, provide the scopes, the endpoints, and the authentication scheme.



IdP Connections | IdP Connection | Browser SSO | Protocol Settings

The metadata below defines how you will interact with this partner using the OpenID Connect protocol. These details may already be complet

SCOPES	address phone openid profile email
AUTHORIZATION ENDPOINT	https://localhost:9031/as/authorization.oauth2
OPENID CONNECT LOGIN TYPE	● CODE ○ FORM POST ○ FORM POST WITH ACCESS TOKEN
AUTHENTICATION SCHEME	○ BASIC ○ POST ○ PRIVATE KEY JWT ● CLIENT SECRET JWT
AUTHENTICATION SIGNING ALGORITHM	HMAC using SHA-256
ENABLE PROOF KEY FOR CODE EXCHANGE (PKCE)	☑

> **Note**
>
> If you clicked **Load Metadata** from the OpenID Provider (OP) on the **General Info** tab, the **Scopes** field and all endpoints are pre-populated, provided that the metadata contains the information.

Field	Description
Scopes	The scopes to be included in the OpenID Connect authentication and OAuth token requests to the OP. Multiple space-separated values are allowed. The default value, without loading metadata from the OP, is `openid`. **Tip** For a list of OpenID Connect defined scopes, see the section about requesting claims using scope values in the OpenID Connection specification at openid.net/specs/openid-connect-core-1_0.html#ScopeClaims.
Authorization Endpoint	The authorization endpoint at the OP. You can enter a relative path, starting with a forward slash, if you provide base URL on the **General Info** tab. There is no default value without loading metadata from the OP.
OpenID Connect Login Type	The OpenID Connect client profile of the client. This client represents PingFederate and is created and managed at the OP. ◦ If the client is configured to support the Basic Client profile, select **Code**. The resulting value of the `response_type` parameter is `code`. * If the client is configured to support the Implicit Client profile, select **Form POST**. The resulting value of the `response_type` parameter is `id_token`. * If the client is configured to support the Implicit Client profile and the target application requires the associated access token, select **Form POST with access token**. The resulting values of the `response_type` parameter are `id_token token`. The default selection, without loading metadata from the OP, is **Code**.
Authentication Scheme	The client authentication method that PingFederate uses. Applicable and visible only to clients supporting the Basic Client profile. ◦ Select **Basic** to submit credentials with HTTP Basic authentication. ◦ Select **POST** to submit credentials with POST. ◦ Select **Private Key JWT** to authenticate with the `private_key_jwt` Client Authentication method. For more information, see Client Authentication in the OpenID Connect specification. ◦ Select **Client Secret JWT** to authenticate with the `client_secret_jwt` Client Authentication method. For more information, see Client Authentication in the OpenID Connect specification. The default selection, without loading metadata from the OP, is **Basic**.

Field	Description
Authentication Signing Algorithm	If **Private Key JWT** or **Client Secret JWT** is the chosen authentication scheme, select the algorithm that PingFederate uses to sign the JSON Web Token (JWT). If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect. ⓘ **Note** If static keys for OAuth and OpenID Connect are enabled, Elliptic-curve cryptography (EC) algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration might affect runtime transactions and require additional changes here. For more information, see Keys for OAuth and OpenID Connect. ⓘ **Note** Based on the chosen signing algorithm, PingFederate selects the signing JSON Web Key (JWK) from its JWK Set (JWKS) at runtime. For the OP to validate the signed JWT, ensure that the OP can access the PingFederate JWKS endpoint, which returns the current JWKS. The PingFederate JWKS endpoint is located at `<Base URL>/pf/JWKS`, where **Base URL** is defined on **System > Server > Protocol Settings > Federation Info**. For example, if the **Base URL** field value is https://www.example.com, the PingFederate JWKS endpoint is https://www.example.com/pf/JWKS. You can pass the PingFederate JWKS endpoint directly to the OP or have the OP contact the PingFederate OP configuration endpoint to obtain the information. For more information, see OpenID Provider configuration endpoint. If **Client Secret JWT** is the chosen authentication scheme, the signing algorithms are `HS256`, `HS384`, and `HS512`.
Enable Proof Key for Code Exchange (PKCE)	Select this check box to enable PingFederate to send a SHA256 code challenge and corresponding code verifier as a Proof Key for Code Exchange (PKCE) to the OP during the Code authentication flow. This check box is applicable and visible only when the **OpenID Connect Login Type** is **Code**. ⓘ **Note** When **Load Metadata** on the **General Info** tab is clicked, PingFederate displays the **Enable PKCE** check box if S256 is listed as a supported method in the `code_challenge_methods_supported` by the OP.

Field	Description
Pushed Authorization Request Endpoint	The Pushed Authorization Request (PAR) endpoint at the OP. When you configure a PAR endpoint, the IdP connection sends authorization requests directly to this endpoint. All parameters associated with an authorization request are transmitted to the PAR endpoint. For more information about the PAR protocol, see OAuth 2.0 Pushed Authorization Requests on the IETF website. You can enter the relative path, `/par`, starting with a forward slash if you provide the base URL on the General Info tab. **Note** If you clicked **Load Metadata** from the OP on the **General Info** tab, the **Pushed Authorization Request Endpoint** field is pre-populated, provided that the metadata contains the information. As such, PAR requests are the new default behavior.
Token Endpoint, **UserInfo Endpoint**, and **JWKS URL**	OAuth 2.0 and OpenID Connect 1.0 endpoints at the OP. For more information, see openid.net/connect. ***Token Endpoint*** The **Token Endpoint** field is only visible and required for clients supporting the Basic Client profile. In other words, the **OpenID Connect Login Type** field is set to **Code**. ***UserInfo Endpoint*** The **UserInfo Endpoint** field is optional. If omitted, PingFederate only has access to the end-user claims from the ID tokens. ***JWKS URL*** The **JWKS URL** is required for PingFederate to validate the inbound ID tokens from the OP. If the OP signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment, or integrated with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. There are no default values without loading metadata from the OP.
Sign Request	Select this check box to send request parameters as claims in a request object, a self-contained, signed JWT as one `request` query parameter to the OP. When this optional configuration is enabled, the OP can validate the integrity of the request parameters based on the digital signature found in the signed JWT. For more information, see the section explaining passing a request object by value in the OpenID Connect specification at openid.net/specs/openid-connect-core-1_0.html#RequestObject. When this optional configuration is enabled, the JWT signed request object includes the `jti` (JWT ID) value. This check box is not selected by default, in which case PingFederate sends request parameters with multiple query parameters, unsigned.

Field	Description
Request Signing Algorithm	Select the algorithm that PingFederate uses to sign the request object. Applicable and visible only when the **Sign Request** check box is selected. If PingFederate is deployed to run in a Java 8 or Java 11 runtime environment, or is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OIDC, additional RSASSA-PSS signing algorithms become available for selection. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect. ⓘ **Note** If static keys for OAuth and OpenID Connect are enabled, Elliptic-curve cryptography (EC) algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration might affect runtime transactions and require additional changes here. For more information, see Keys for OAuth and OpenID Connect. ⓘ **Note** PingFederate automatically selects the signing JSON web key (JWK) based on the selected signing algorithm from its JWK Set (JWKS). In order for the OP to validate the signed request object, ensure that the OP can access your PingFederate's JWKS URL, which returns the current set of JSON web keys. The PingFederate JWKS URL is located at `<Base URL>/pf/JWKS` , where **Base URL** is defined on **System > Server > Protocol Settings > Federation Info**. For example, if the **Base URL** field value is https://www.example.com, the PingFederate JWKS URL is https://www.example.com/pf/JWKS. You can pass the JWKS URL directly to the OP or have the OP contact the PingFederate OpenID Provider configuration endpoint for it. For more information, see OpenID Provider configuration endpoint.
Track User Sessions for Logout	When selected, PingFederate tracks logout entries in the user session so that PingFederate can handle back-channel logout requests. Also, when selected, the **IdP Connection** window's **Summary & Activation** tab displays the connection's **Back-Channel Logout URI**. The check box is cleared by default.

+ . **Optional:** Remain on the **OpenID Provider Info** tab and specify the request parameters that are allowed to be included in the authentication requests to the OP under **Request Parameters**. For more information, see Configuring request parameters and SSO URLs.

1. **Optional:** On the **Overrides** tab, specify a default target URL and authentication context overrides.

2. On the **Activation & Summary** tab, review your connection settings.

   When you finish setting up a connection, you can choose to activate it immediately.

> **Important**
>
> Regardless of whether you choose to activate a new connection now or later, you must click **Save** on the **Summary & Activation** tab for a new connection if you want to keep the configuration.

You can deactivate a connection at any time. When a connection is inactive, all transactions to or from this partner are disabled.

In this use case, because PingFederate is an OAuth client, you are likely required by the authorization server at the OP to register the **Redirect URI** as shown on the **Summary & Activation** tab. This registration should be associated with the client that represents PingFederate, the client that you have provided on the **General Info** tab. For more information, see the documentation provided by the OP.

The **SSO Application Endpoint** provides a sample URL at the `/sp/startSSO.ping` application endpoint that webmasters or web application developers at your site can use to invoke single sign-on (SSO) for the connection. For a list of supported parameters, see Viewing SP application endpoints.

If you have selected the **No Mapping** option on the **Identity Mapping** tab, the **Summary & Activation** tab does not show the **SSO Application Endpoint** sample URL.

The target application can also invoke SSO requests by contacting the `/sp/init_login.ping` SP protocol endpoint. For more information, see Configuring request parameters and SSO URLs.

Configuring request parameters and SSO URLs

On the **OpenID Provider Info** tab of the **Protocol Settings** window, you can define request parameters under **Request Parameters**.

About this task

You can define request parameters for the following purposes:

- Allow custom request parameters to be include in the authentication requests to support OpenID provider (OP)-specific use cases.
- Define the default values for the request parameters.
- Specify whether the default values, if any, can be overridden at runtime.
- Allow the target application to request different scopes at runtime. The OP can reject the requested scopes based on its client configuration.
- Protect the requested authentication request, `acr_values`, the authentication requirement, `prompt`, or both so that none of them can be overridden at runtime by the application endpoint parameters - `RequestedAuthnCtx`, `IsPassive`, and `ForceAuthn`.

Direct mapping is available for types of parameters with a contract or text value:

- Client ID
- Client IP
- Extended properties
- SP Connection Entity ID
- OAuth scopes
- Tracked HTTP parameters

Use the No Mapping parameter type to capture "no default value".

You can use the following types of parameters in expressions:

- Signed request object claims
- Chained attributes
- SAML authentication request
- HTTP Request

Steps

1. Enter the request parameter's **Name**.
2. Select the request parameter's **Type**
   - **Context**
   - **Expression**
   - **Extended Properties**
   - **No Mapping**
   - **Text**
   - **Tracked HTTP Parameters**
3. Enter the request parameter's default **Value**.

   If you selected the **Context** type, the following are available for direct mapping:

   - **Client ID**
   - **Client IP**
   - **OAuth scopes**
   - **SP Connection Entity ID**

     This is optional if the target application is allowed to override the parameter value at runtime. When no default value is specified, any value provided by the target application is accepted by the `/sp/startSSO.ping` service provider (SP) application endpoint and the authentication endpoint. If the target application does not provide the parameter in its single sign-on (SSO) URL and no default value is specified, the parameter is not included in the authentication requests.

     This is required if the target application is not allowed to override the parameter value at runtime.

   When specified, the request parameter is always included in the authentication requests. If the target application is not allowed to override the parameter value at runtime, the default value is sent.
4. **Optional:** To let the target application override the request parameter's default value during runtime, select the **Application Endpoint Override** check box.

When this check box is selected, during runtime:

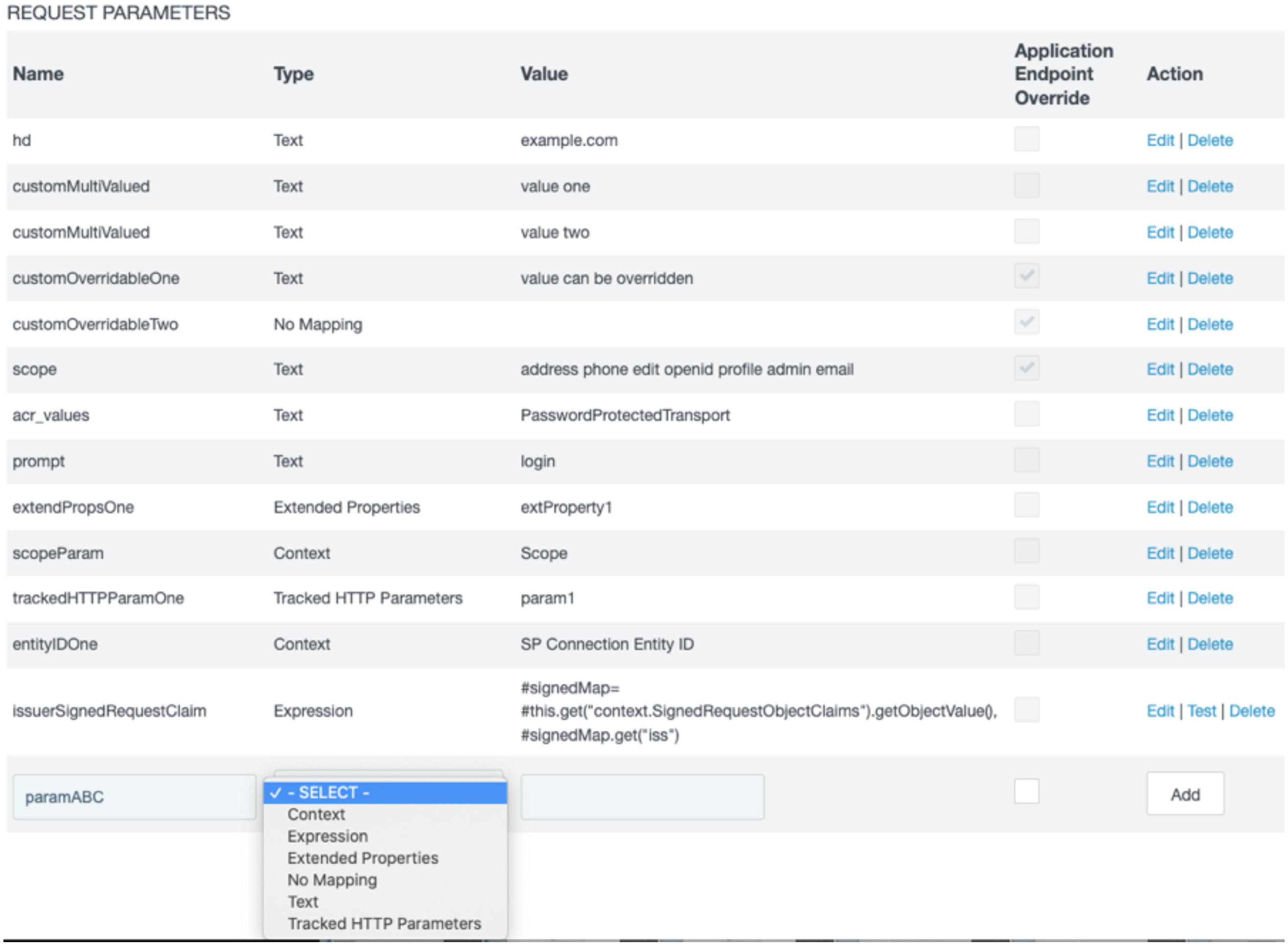
- If the target application provides the parameter in its SSO URL to the `/sp/startSSO.ping` SP application endpoint or the authentication endpoint, the value in the SSO URL is used.
- If the target application does not provide the parameter in its SSO URL, the default value is used. However, if the parameter doesn't have a default value, then the parameter is not included in authentication requests.

> **Note**
>
> The `/sp/init_login.ping` SP endpoint does not accept overridden values. The `login_hint` parameter is the only exception. The default value, if any, is used. See the note at the end of this topic for more information.

5. Click **Add**.

Example

Consider the following examples of request parameters:



REQUEST PARAMETERS

Name	Type	Value	Application Endpoint Override	Action
hd	Text	example.com	☐	Edit | Delete
customMultiValued	Text	value one	☐	Edit | Delete
customMultiValued	Text	value two	☐	Edit | Delete
customOverridableOne	Text	value can be overridden	☑	Edit | Delete
customOverridableTwo	No Mapping		☑	Edit | Delete
scope	Text	address phone edit openid profile admin email	☑	Edit | Delete
acr_values	Text	PasswordProtectedTransport	☐	Edit | Delete
prompt	Text	login	☐	Edit | Delete
extendPropsOne	Extended Properties	extProperty1	☐	Edit | Delete
scopeParam	Context	Scope	☐	Edit | Delete
trackedHTTPParamOne	Tracked HTTP Parameters	param1	☐	Edit | Delete
entityIDOne	Context	SP Connection Entity ID	☐	Edit | Delete
issuerSignedRequestClaim	Expression	#signedMap= #this.get("context.SignedRequestObjectClaims").getObjectValue(), #signedMap.get("iss")	☐	Edit | Test | Delete
paramABC	✓ - SELECT -		☐	Add

- The `hd` parameter is defined with a default value that cannot be overridden at runtime. The parameter is always included in the authentication requests and the value is always `example.org`.
- The `customMultiValued` parameter is defined with two default values that cannot be overridden at runtime. This multivalued parameter is always included in the authentication requests. The values are always as defined.

- The `customOverridableOne` parameter is defined with a default value that can be overridden at runtime. This parameter is always included in the authentication requests. If the target application provides the parameter in its SSO URL, the value in the SSO URL is used. If the target application does not provide the parameter in its SSO URL, the default value is used.
  - To override the value, configure the target application to append the request parameter and the desired value to the **SSO Application Endpoint**, as shown on the **Summary & Activation** tab, as in the following example.

    https://sso.example.com/sp/startSSO.ping?PartnerIdpId=https%3A%2F%2Fsso.alpha.local%3A9031 and **customOverridableOne=foo**
  - To construct a multivalued request parameter, append the request parameter multiple times with different values, as in the following example.

    https://sso.example.com/sp/startSSO.ping?PartnerIdpId=https%3A%2F%2Fsso.alpha.local%3A9031 and `customOverridableOne=foo` and `customOverridableOne=bar`

  https%3A%2F%2Fsso.alpha.local%3A9031 is the URL-encoded value of https://sso.alpha.local:9031, the issuer value of the OP.
- The `customOverridableTwo` parameter is defined without a default value. Parameters without default values require the **No Mapping** parameter type. Any value provided by the target application in the SSO URL is accepted. To include this parameter in the authentication requests to the OP, configure the target application to append the request parameter and the desired value to the **SSO Application Endpoint**.
  - To construct a multivalued request parameter, append the parameter multiple times with different values.

    If the target application does not provide the parameter in its SSO URL, the parameter is not included in the authentication requests.
- The `scope` (standard) parameter is defined with a value matching that of the **Scopes** field, on the same tab, and with the option to allow the target application to override the value at runtime. In essence, the target application is allowed to dynamically change the scope it requires at runtime by appending the `scope` parameter and the desired scopes to the **SSO Application Endpoint**.

  > **Note**
  >
  > While the target application can request different scopes, the OP can reject the requested scopes based on its client configuration. Work with the OP to understand which scopes are applicable to your use case to prevent runtime errors.
- The `acr_values` (standard) parameter is defined with a default value that cannot be overridden at runtime. As a result, the `RequestedAuthnCtx` parameter, if supplied in the SSO URL by the target application, is ignored. In the authentication requests, the value of the `acr_values` parameter is always set to the default value specified in the configuration. Define the `acr_values` parameter if you want to protect the requested authentication context from the target application.
- The `prompt` (standard) parameter is defined with a default value of `login` that cannot be overridden at runtime. As a result, the target application will not be able to suppress the reauthentication requirement by including `IsPassive=true` in the SSO URL. In the authentication requests, the value of the `prompt` parameter is always set to `login`.

  Similarly, if the `prompt` parameter is defined with a default value of `none` that cannot be overridden at runtime, the target application will not be able to request the end users to reauthenticate by including `ForceAuthn=true` in the SSO URL. In the authentication requests, the value of the `prompt` parameter is always set to `none`.

- The `issuerSignedRequestClaim` parameter uses an OGNL expression to extract the `iss` claim from an incoming signed request.

> **Note**
>
> These examples use the `/sp/startSSO.ping` SP application endpoint. You can also use the `/sp/init_login.ping` SP protocol endpoint to invoke the Third Party Initiated Login flow. For more information, see View SP protocol endpoints.

> **Important**
>
> For information about URL encoding, see third-party resources, such as HTML URL-encoding Reference.

Query parameters versus request object

By default, PingFederate sends all request parameters through multiple query parameters, unsigned.

If the **Sign Request** check box is selected, PingFederate creates a signed JSON web token (JWT) that contains claims representing the request parameters and passes the signed JWT as one query parameter, `request`, to the OpenID provider (OP). The `client_id`, `response_type`, and `scope` request parameters are always passed to the OP as individual query parameter as well.

Consider the following authentication requests based on the previous sample configuration. The client authenticates through the HTTP Basic authentication scheme and initiates single sign-on (SSO) request without providing overrides for any request parameters.

Request parameters via query parameters

```
https://sso.alpha.local:9031/as/authorization.oauth2
?acr_values=PasswordProtectedTransport
&customMultiValued=value+one
&customMultiValued=value+two
&customOverridableOne=value+can+be+overridden
&hd=example.org
&prompt=login
&nonce=ykulMjpwAFk79R1rBOBWm5
&redirect_uri=https://www.example.com/sp/eyJpc3MiOiJodHRwczpcL1wvc3NvLmFscGhhLmxvY2FsOjkwMzEifQ/cb.openid
&state=e75nIlVU6Wa5TMmOwegDPSEI2iO9zd
&client_id=RP
&response_type=code
&scope=address+phone+edit+openid+profile+admin+email
```

Request parameters via a request object by value

```
https://sso.alpha.local:9031/as/authorization.oauth2
?request=eyJhbG...ZTMifQ.eyJhdW...lJQIn0.IAOpuf...IqCftg
&client_id=RP
&response_type=code
&scope=address+phone+edit+openid+profile+admin+email
```

> **Note**
>
> The `client_id`, `response_type`, and `scope` request parameters are always passed to the OP as individual query parameters as defined in the OpenID Connect specification.

The value of the `request` query parameter, truncated for readability, is the request object, a signed JWT that contains the request parameters as individual claims, illustrated in the following decoded payload.

```
{
  "aud": "https://sso.alpha.local:9031",
  "exp": 1495645410,
  "acr_values": "PasswordProtectedTransport",
  "customMultiValued": [
    "value one",
    "value two"
  ],
  "customOverridableOne": "value can be overridden",
  "hd": "example.org",
  "prompt": "login",
  "nonce": "vhW2VJc7eZ6r6vfpiAwepd",
  "redirect_uri": "https://sso.rp.local:9021/sp/eyJpc3MiOiJodHRwczpcL1wvc3NvLmFscGhhLmxvY2FsOjkwMzEifQ/cb.openid",
  "state": "nFVzgFirZtg3kBXMFpWt5RNhO4oDuA",
  "client_id": "RP",
  "response_type": "code",
  "scope": "address phone edit openid profile admin email"
}
```

For more information, see the section explaining passing a request object by value in the OpenID Connect specification at openid.net/specs/openid-connect-core-1_0.html#RequestObject.

Configuring IdP discovery using a persistent cookie

PingFederate's proprietary identity provider (IdP)-discovery method makes use of an IdP persistent reference cookie (IPRC) to track the identity provider with whom a user last authenticated.

About this task

There are three significant differences between standard IdP discovery and the IPRC method:

- Standard IdP discovery can be used only with SAML 2.0, but the IPRC can be used with any federation protocol.
- The common domain cookie (CDC) can be configured as a temporary, session-based cookie. The IPRC always persists for a configurable period of time.
- The CDC is set by the IdP and is readable by both federation partners. The IPRC is set by the service provider (SP), using information in the SAML assertion, and cannot be accessed by the IdP.

The deployed connection configuration between SP and IdP partners must include SP-initiated single sign-on (SSO).

Steps

1. Edit the `org.sourceid.websso.profiles.sp.IdpIdCookieSupport.xml` file located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

2. Set the value of `EnableIdpIdCookie` to `true`.

3. **Optional:** Modify the remaining elements in the configuration, as described in the following table.

Field	Description
IdpIdCookieName	The name of the IPRC set by the SP installation. The default is `IdPId`. The cookie name cannot contain any of the following characters: `&`, `>`, `<`, `;`, a comma, or a space.
IdpIdCookieLifeTimeInDays	The lifetime for the cookie. The default is `365` days and a maximum of `24855` days. The browser will delete the cookie when the period is expired.
ShowIdpSelectionList	If set to `true`, the default, the SP displays a list of IdPs that can be used to initiate the SSO event if the cookie is not set. If set to `false`, the SP installation generates an error page.

4. Start or restart PingFederate.

> **Note**
>
> After an IPRC cookie is set, the only way to change the IdP to whom the SP will send Authentication Requests for the user is to do one of the following: wait for the cookie to expire, delete the cookie, or perform IdP-initiated SSO using the new IdP.

System administration

This section describes general administrative functions for PingFederate.

Configuring PingFederate properties

The default administrative console and runtime behavior of PingFederate is controlled in part by configuration properties set in the `run.properties` file, located in the `<pf_install>/pingfederate/bin` directory.

About this task

The most common properties are documented in the following table. For the rest of the properties, including various cookie-encoding options, see the `run.properties` file.

> **Tip**
>
> The clustering configuration options are also maintained in the `run.properties` file. For more information, see Deploying cluster servers.

Property	Description
`pf.admin.https.port`	Defines the port on which the PingFederate administrative console runs. The default value is `9999`.

Property	Description
`pf.admin.baseurl`	Defines the URL that PingFederate's administrative node uses to populate resource references in Administrative application programming interface (API) responses. The administrative node also uses it for the redirect URL it sends to an OpenID Provider (OP) for administrator OpenID Connect (OIDC) (for example, https://pingfederate-admin.example.com or, if the load balancer uses a custom port, https://pingfederate-admin.example.com:8443). The default value is blank. Use `pf.admin.baseurl` instead of `pf.admin.hostname`, which has been deprecated. If `run.properties` defines both, PingFederate ignores `pf.admin.hostname`. But if `run.properties` defines only `pf.admin.hostname`, PingFederate constructs the URL the same way it does in versions of PingFederate before 10.3.
`pf.console.bind.address`	Defines the IP address over which the PingFederate administrative console communicates. Use for deployments where multiple network interfaces are installed on the machine running PingFederate.
`pf.console.title`	Defines the browser window or tab title for the administrative console. It makes separate instances easily identifiable.
`pf.console.environment`	Defines the name of the PingFederate environment that will be displayed in the administrative console. It makes separate environments easily identifiable.
`pf.console.show.background.images`	Enables or disables the background images on the dashboard of the administrative console. The images are enabled by default.
`pf.console.session.timeout`	Defines the length of time in minutes until an inactive administrative console times out. The minimum setting is 1 minute, and maximum is 8 hours (480 minutes). Default is `30` minutes.
`pf.log.eventdetail`	Enables or disables (the default) detailed event logging for actions performed by administrative console users.
`pf.console.login.mode`	Indicates whether more than one administrative user may access the administrative console at one time. Supported values: Single | Multiple. The default value is `Multiple`.
`pf.console.authentication`	Indicates whether administrators sign on to PingFederate using credentials managed internally by PingFederate or externally by other systems.

Property	Description
`pf.admin.api.authentication`	Defines the authentication method of the PingFederate administrative API. Valid values are: • `none` - No direct login method is available. • `native` - Internal password file authentication. • `LDAP` - External LDAP authentication. • `cert` - X509 certificate-based authentication. • `RADIUS` - External RADIUS authentication. • `OAuth2` - External or internal OAuth2 authorization. The default value is `native`. The values are case-insensitive. You can also configure PingFederate to support both `OAuth2` authorization and a basic authentication method by specifying two values separated with a comma. For example, specify `pf.admin.api.authentication=OAuth2,LDAP`. The basic authentication methods are `native`, `LDAP`, and `RADIUS`. Supporting two authentication methods is helpful when you want to change applications from one method to another. **Note** When configuring support for two authentication methods, consider the following: • The order of the values is not important. PingFederate uses the HTTP Authorization request header to determine the authorization scheme. A request can contain only one Authorization header. • You cannot combine the `none` and `cert` values with other values. • If you specify an invalid value or more than two values, PingFederate will fail on startup. **Caution** You should only allow multiple authentication sources while migrating from one source to another because using multiple authentication sources simultaneously increases security risks. For administrative API authentication, after migration, you should use `OAuth2` for the authentication source.

Property	Description
`pf.pingone.admin.url.region`	These properties set the URL of the PingOne unified admin icon in the PingFederate administrative console. This property should be set based on the region of your PingOne organization. Choose one of the following region-specific values for your environment. ***com*** console.pingone.com ***eu*** console.pingone.eu ***asia*** console.pingone.asia ***ca*** console.pingone.ca
`pf.pingone.admin.url.environment.id`	Defines the ID of your PingOne organization's environment.
`ldap.properties.file`	When Lightweight Directory Access Protocol (LDAP) administrative console authentication is enabled, indicates the name of the file containing configuration properties.
`cert.properties.file`	When certificate-based console authentication is enabled, indicates the name of the file containing configuration properties.
`radius.properties.file`	When RADIUS-based console authentication is enabled, indicates the name of the file containing configuration properties.
`oidc.properties.file`	When OIDC administrative-console authentication is enabled, indicates the name of the file containing configuration properties.
`pf.http.port`	Defines the port on which PingFederate listens for unencrypted HTTP traffic at runtime. For security reasons, this port is turned off by default. **Caution** This port should remain disabled in production if your deployment configuration directly exposes the PingFederate server to the Internet.
`pf.https.port`	Defines the port on which PingFederate listens for encrypted HTTPS (SSL/TLS) traffic. The default value is `9031`.

Property	Description
`pf.secondary.https.port`	Defines a secondary HTTPS port that can be used for mutual SSL/TLS (client X.509 certificate) authentication for both end users and protocol requests (Security Assertion Markup Language (SAML), WS-Trust, and OAuth). Set its value to the desired inbound listening TCP port. A value of `-1` disables this feature. **Important** If you are using client X.509 certificates for either WS-Trust Security Token Service (STS) authentication or for SAML back-channel authentication, you must use this port, or a similarly configured new listener, with either the `WantClientAuth` or `NeedClientAuth` parameter set to `true` in the `jetty-runtime.xml` file. For more information, see the note at the end of this table.
`pf.engine.bind.address`	Defines the IP address over which the PingFederate server communicates with partner federation gateways. Use for deployments where multiple network interfaces are installed on the machine running PingFederate.
`pf.monitor.bind.address`	Defines the IP address over which Java Management Extensions (JMX) communicate with PingFederate. Use for deployments where multiple network interfaces are installed on the machine running PingFederate.
`pf.engine.prefer_ipv4`	Defines the protocol to be used by PingFederate. `True`, the default, enables use of IPv4 only. `False` enables use of both IPv4 and IPv6.
`http.proxyHost` and `http.proxyPort`	Specifies the hostname, or the IP address, and the port number of the forward proxy server that HTTP traffic originating from PingFederate must go through.
`https.proxyHost` and `https.proxyPort`	Specifies the hostname, or the IP address, and the port number of the forward proxy server that HTTPS traffic originating from PingFederate must go through.
`http.nonProxyHosts`	Specifies one or more destinations where PingFederate is not required to proxy its HTTP and HTTPS traffic through the forward proxy server configured by the `http[s].proxyHost` and `http[s].proxyPort` properties. This property supports multiple values separated by the pipe character (`|`) and the wildcard character (`*`) for pattern matching. See the example below. `*.example.com|localhost`

Property	Description
`pf.runtime.context.path`	Allows customization of the server path for PingFederate endpoints. ⓘ **Note** If this property is changed, the path must also be added to the base URL for your PingFederate environment. Base URL is defined on **System > Server > Protocol Settings > Federation Info**. The `pf.runtime.context.path` property is also compatible with virtual host names. Unlike the base URL configuration, the virtual host names configuration does not require any context path. Virtual host names are defined on **System > Server > Virtual Host Names**. For example, suppose the base URL is https://www.example.com:9031 and the virtual host names are \www.example.org and \www.example.info. To configure the `pf.runtime.context.path` property value as `/sso`, you must update the base URL to https://www.example.com:9031/sso but leave the virtual host names as they are. Once configured, you can access the runtime server at the following endpoints: ***Base URL*** • https://www.example.com:9031/sso ***Virtual host names*** • https://www.example.org:9031/sso • https://www.example.info:9031/sso
`pf.runtime.http.maxRequestSize`	Sets the maximum request size in bytes for inbound runtime requests. The default value is `200000` bytes.
`pf.log.dir`	Network path to the output location of log files. The default is `<pf_install>>/pingfederate/log`
`pf.hsm.mode`	Enables or disables (the default) a FIPS-compliance Hardware Security Module (HSM).
`pf.hsm.hybrid`	Enables or disables the HSM hybrid mode. Applicable only when the `pf.hsm.mode` property is configured to use an HSM. When set to `true`, keys and certificates can be stored on either the HSM or the local trust store. When set to `false`, the default setting, keys and certificates are stored on the HSM when applicable. The HSM hybrid mode allows an organization to move the storage of keys and certificates from the local trust store to an HSM over time without deploying a new PingFederate installation and mirroring the setup. For more information, see Transitioning to an HSM.

Property	Description
`org.bouncycastle.fips.approved_only`	When the `pf.hsm.hybrid` property is set to `true`, this property can be set to `true` or `false`. In this case, the recommended setting is `false`. If `pf.hsm.hybrid` is set to `false`, this property must be set to `true`. In FIPS-approved mode only, the module will provide approved algorithms only. For more information, see https://www.bouncycastle.org/fips-java. The default setting is `true`.
`pf.provisioner.mode`	Enables or disables (the default) outbound provisioning. Also used to enable provisioning failover.
`pf.heartbeat.system.monitoring`	Enables or disables (the default) the heartbeat endpoint, `/pf/heartbeat.ping`, to return detailed system monitoring information through a customizable Velocity template file . For more information, see Customizing the heartbeat message. When set to `false`, the `/pf/heartbeat.ping` endpoint returns OK.When set to `true`, the `/pf/heartbeat.ping` endpoint returns all available stats.
`org.apache.xml.security.ignoreLineBreaks`	Determines whether PingFederate omits line breaks in XML digital signatures. If omitted, this setting defaults to `false`. Set this property to `true` for improved interoperability with Microsoft products.
`sun.net.client.defaultConnectTimeout`	Determines the default connect timeout for outbound java.net.URL connections in milliseconds. The default setting is 10000.
`sun.net.client.defaultReadTimeout`	Determines the default read timeout for outbound java.net.URL connections in milliseconds. The default setting is 10000.
`pf.admin.threads.min`	The minimum number of threads in HTTP server thread pools for the administrative console. The default value is `1`.
`pf.admin.threads.max`	The maximum number of threads in HTTP server thread pools for the administrative console. The default value is `10`.
`pf.runtime.threads.min`	The minimum number of threads in HTTP server thread pools for the runtime engine nodes. The default value is `10`.
`pf.runtime.threads.max`	The maximum number of threads in HTTP server thread pools for the runtime engine nodes. The default value is `200`.
`pf.admin.acceptQueueSize`	The queue size of the HTTP connector for the administrative console. The default value is `512`.
`pf.runtime.acceptQueueSize`	The queue size of the HTTP connector for the runtime engine nodes. The default value is `512`.

Note

Additional configuration of the listener ports, including adding new listeners, is available through the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file. For example, options include the `WantClientAuth` and `NeedClientAuth` flags, which indicate that a client certificate is either requested or required, respectively, for mutual SSL/TLS. For the pre-configured SSL secondary port, the `WantClientAuth` parameter is set to `true` and the `NeedClientAuth` parameter is set to `false` by default.

Steps

1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.

   You should consider creating a backup copy of the file.

2. Modify the applicable properties.

3. Restart PingFederate.

Result

Important

You must manually configure the runtime server-related properties on each engine node. The `run.properties` file is not copied from the console node to the engine nodes automatically. Also, it is not part of the **Replicate Configuration** process. If running, restart PingFederate.

Overriding configuration settings using environment variables

To change a PingFederate server’s configuration, you can use environment variables to override the settings in multiple configuration files instead of modifying those files.

If you use an environment variable to override a configuration file setting, you don’t need to remove the setting from the configuration file. If a setting exists in both a configuration file and an environment variable, the environment variable always takes precedence.

You can use a script or command line to assign values to environment variables. Then when PingFederate starts, it overrides the values in the configuration files with the values of the environment variables. During startup, PingFederate logs all the environment variables that start with "`PF_`" in the `init.log`.

You can override any property value in the following files:

`<pf_install>/pingfederate/bin/`

- `*.properties`

`<pf_install>/pingfederate/server/default/conf/`

- `*.properties`
- `*.conf`

Important

- In clustered PingFederate environments, you must apply the environment variable changes to all the nodes.
- Because the configuration settings overridden by the environment variables also apply to the scripts under `<pf_install>/pingfederate/bin/` , prevent misconfiguration by ensuring you set the environment variables correctly before running those scripts or else using persistent environment variables.

The format of an environment variable is `PF_<FILE_NAME_WITHOUT_EXTENSION>_<property_name>` .

The filename part of the variable is uppercase. The property name part is lowercase and case-sensitive for properties handled by both PingFederate and external libraries like jetty.

Here are some examples of environment variables:

The environment variable for the `pf.admin.https.port` property in the `run.properties` file is `PF_RUN_pf_admin_https_port` .

In a Unix-based environment, the following command sets the value of that environment variable to `9998` :

```
export PF_RUN_pf_admin_https_port=9998
```

The environment variable for the `track.state` property in the `cluster-adaptive.conf` file is `PF_CLUSTER_ADAPTIVE_track_state` .

In a Unix-based environment, the following command sets the value of that environment variable to `false` :

```
export PF_CLUSTER_ADAPTIVE_track_state=false
```

Note

The `service-points.conf` file in the `<pf_install>/pingfederate/server/default/conf/` directory has replaced the `hivemodule.xml` file that was in the `<pf_install>/pingfederate/server/default/conf/META-INF/` directory. This lets you use environment variables to also override implementation classes for core services in PingFederate. For more information, see "Replacement of hivemodule.xml" in the Upgrade considerations.

Configuring size limits

You can configure size limits, which include values in the IdP Session Registry, the SP Session Registry, the Inter-Request State management service, and others.

Steps

1. To configure a size limit, edit the `<pf_install>/pingfederate/server/default/conf/size-limits.conf` file.

2. Modify the applicable settings. The setting are described in the following table.

Setting	Description
The IdP Session Registry stores SLO-related session information for IdP adapters, as well as for IdP and SP connections that a user has interacted with. It also stores sessions for PingFederate’s Authentication Sessions feature.	
IdpSessionRegistryMapImpl.max.sessions	This setting controls the maximum number of user sessions (for SLO or Authentication Sessions) kept in memory. When this limit is reached, sessions are removed on a least-recently-used basis. The default setting is 10000.
IdpSessionRegistryMapImpl.max.individual.sessions	This setting controls the maximum number of IdP adapter or IdP connection sessions per user session. When this limit is reached, sessions are removed on a first-in first-out basis. The default setting is 500.
IdpSessionRegistryMapImpl.max.partner.sessions	This setting controls the maximum number of SP connection sessions per IdP adapter or IdP connection session. When this limit is reached, sessions are removed on a first-in first-out basis. The default setting is 500.
IdpSessionRegistryMapImpl.max.user.keys	This setting defines the maximum number of unique user keys that can be tracked. When this limit is reached, user keys are removed on a least-recently-used basis. The default setting is 50000.
IdpSessionRegistryMapImpl.max.user.key.sris	This setting defines the maximum number of SRIs (browser sessions) that can be tracked for a given unique user key. When this limit is reached, SRIs are removed and revoked on a least-recently-used basis. The default setting is 100.
IdpSessionRegistryMapImpl.expiry.mins	This setting defines the expiry period for user sessions in minutes. If no activity has been seen for a given user session for this period, it will be removed. The default setting is 1440
The SP Session Registry stores SLO-related session information for SP adapters and IdP connections that a user has interacted with.	
SpSessionRegistryMapImpl.max.sessions	This setting controls the maximum number of user sessions kept in memory. When this limit is reached, sessions are removed on a least-recently-used basis. The default setting is 10000.

Setting	Description
SpSessionRegistryMapImpl.max.individual.sessions	This setting controls the maximum number of SP adapter sessions per user session. When this limit is reached, sessions are removed on a first-in first-out basis. The default setting is 500.
SpSessionRegistryMapImpl.expiry.mins	This setting defines the expiry period for user sessions in minutes. If no activity has been seen for a given user session for this period, it will be removed. The default setting is 1440.
The Inter-Request State Management service has two maps. The 'state' map is used to store short-lived state information between requests within an SSO or SLO transaction. The 'attr' map is used by adapters (such as the HTML form adapter) to store user session attributes.	
InterReqStateMgmtMapImpl.max.size.state.map	This setting controls the maximum number of user sessions in the state map. When this limit is reached, sessions are removed on a least-recently-used basis. The default setting is 10000.
InterReqStateMgmtMapImpl.expiry.mins.state.map	This setting controls the expiry period for user sessions in the state map. If no activity has been seen for a given user session for this period, it will be removed. The default setting is 30.
InterReqStateMgmtMapImpl.max.size.attr.map	This setting controls the maximum number of user sessions in the attribute map. When this limit is reached, sessions are removed on a least-recently-used basis. The default setting is 10000.
InterReqStateMgmtMapImpl.expiry.mins.attr.map	This setting controls the expiry period for user sessions in the attribute map. If no activity has been seen for a given user session for this period, it will be removed. The default setting is 1440.
InterReqStateMgmtMapImpl.max.session.attrs	This setting controls the maximum number of attributes stored in the attribute map for a given user session. The default setting is 500.
SessionRevocationServiceMapImpl.max.revoked.sris	This setting controls the maximum number of revoked session identifiers kept in memory. When this limit is reached, revoked identifiers are removed on a first-in first-out basis. The default setting is 50000.

Setting	Description
MetadataDirectory.max.size.idp.conn.map	This setting controls the maximum number of IdP connections kept in memory. When this limit is reached, connections are removed on a least-recently-used basis. The default setting is 10000.
MetadataDirectory.max.size.sp.conn.map	This setting controls the maximum number of SP connections kept in memory. When this limit is reached, connections are removed on a least-recently-used basis. The default setting is 10000.
ClientManagerXmlFileImpl.max.size.clients.map	This setting controls the maximum number of OAuth clients kept in memory. When this limit is reached, clients are removed on a least-recently-used basis. The default setting is 10000.

3. Restart PingFederate.

PingFederate log files

PingFederate records document server events depending on your configuration preferences.

PingFederate generates these logs that document server events:

`admin.log`

Records actions performed by administrative console users.

`admin-event-detail.log`

Records detailed information about each applicable administrative console event performed by administrative console users if detailed event logging is enabled.

`admin-api.log`

Records actions performed by administrative-API users.

`runtime-api.log`

Records actions performed by API users using the OAuth Client Management Service, the OAuth Access Grant Management Service, and the Session Revocation API.

`transaction.log`

Records individual identity-federation runtime transactions at specified levels of detail.

`audit.log`

Records a selected, configurable subset of transaction log information plus additional details, intended for security-audit and regulatory compliance purposes.

`provisioner-audit.log`

Records outbound provisioning events, intended for security-audit purposes.

`provisioner.log`

Records only provisioning activity.

`server.log`

Records PingFederate runtime and administrative server activities.

`init.log`

Records only Jetty messages generated prior to PingFederate start up.

These log files are written to the PingFederate log directory. The default location is the `<pf_install>/pingfederate/log` directory. As needed, administrators can change the log directory by modifying the `pf.log.dir` property in the `<pf_install>/pingfederate/bin/run.properties` file.

Log4j 2 logging service and configuration

PingFederate uses the Log4j 2 logging service to generate its log files.

Configurations are maintained in the `log4j2.xml` file, located in the `<pf_install>/pingfederate/server/default/conf` directory.

> **Note**
>
> The `log4j2.xml` configuration file is individually managed per PingFederate server. This flexibility allows multiple PingFederate nodes to write different level of messages to different destinations.
> If you want all PingFederate servers to use the same logging configuration, manually synchronize the `log4j2.xml` file across multiple PingFederate servers.

Log levels and verbosity

Log messages are categorized into six log levels:

1. `FATAL`
2. `ERROR`
3. `WARN`
4. `INFO`
5. `DEBUG`
6. `TRACE`

PingFederate only records messages tagged with log level `INFO`, `WARN`, `ERROR`, and `FATAL` to the server log and the provisioner log. Messages with `DEBUG`, or `TRACE` tags, are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purposes, you can enable console logging or verbose messages.

> **Important**
>
> When you no longer require console logging or verbose messages, turn them off. On Windows, never highlight the console output because it might slow or stop PingFederate from processing requests.

For the audit log, the provisioner audit log, and the transaction log, any setting lower than `INFO` (`WARN` , `ERROR` , or `FATAL`) turns logging off.

For more information, see Enabling debug messages and console logging.

PingFederate activates the changes within half a minute. You don’t need to restart PingFederate.

Fields (and attributes)

You can customize some logs, such as the audit log and the administrative API log, to log additional or less information by modifying their `pattern` elements. The `log4j2.xml` file documents available fields inline.

> **Tip**
>
> You can configure PingFederate to log user attributes, if they are present, in the audit log, transaction log, and server log. When you require privacy for sensitive user attributes, select the corresponding check boxes under **Mask Log Values** to mask their values in these logs.

In addition, messages in the audit log and the server log are recorded with a tracking ID, which can be used to identify subsequent, related transactions. The tracking ID can be used for troubleshooting and support purposes, to aggregate and analyze log entries tied to the same original request. The tracking ID, `%X{trackingid}` , can also be added to the configuration for the transaction log, or removed from the audit log and the server log by modifying the `pattern` element for the logs in the `log4j2.xml` configuration file.

Log formats

The audit log and the provisioner audit log can be written in Common Event Format (CEF). Furthermore, the audit log can also be written in a format used in conjunction with Splunk and the Splunk App for PingFederate. The `log4j2.xml` file comes preset with configuration samples to ease the setup.

Log destinations

The audit log, the provisioner audit log, the provisioner log, and the server log can be written to databases. PingFederate installation includes setup scripts for various tables, located in the `<pf_install>/pingfederate/server/default/conf/log4j/sql-scripts` directory, and configuration samples in the `log4j2.xml` file.

Log rotation

Most PingFederate-generated log files roll over at midnight each day. The system keeps all of the resulting historical log files. Some log files, such as the `audit.log` file, the `audit-event-detail.log` file (if enabled), the `provisioner-audit.log` file (when applicable), and the `transaction.log` , can become quite large, depending on your production load and settings. You might want to back up or remove older files on a routine basis.The `server.log` file is rolled over when it reaches 10 MB. Five old log files are kept before the oldest file is removed. Administrators can adjust the file size and the number of files to be retained in the `log4j2.xml` configuration file, as needed.

For more information about Log4j 2, see the Log4j 2 open-source project.

HTTP request logging

HTTP requests to the runtime engine and the administrative console are logged to the `<date>.request.log` file and `<date>.request2.log`, respectively, by the Pingfederate web container.

Like other PingFederate-generated log files, the HTTP request logs are written to the default PingFederate log directory. Properties controlling request logging are contained in the web-container configuration files:

- `jetty-runtime.xml` for the runtime engine (the `<date>.request.log` files)
- `jetty-admin.xml` for the administrative console (the `<date>.request2.log` files)

You can find these files in the `<pf_install>/pingfederate/etc` directory, and you can independently manage them on a per-server basis.

Administrator audit logging

PingFederate records actions performed by server administrators.

This information is recorded in the `<pf_install>/pingfederate/log/admin.log` file. The events themselves are not configurable, but you can adjust Log4j 2 configuration settings to deliver the desired level of detail surrounding each event in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

Events logged by PingFederate include but are not limited to:

- Sign on attempt
- Explicit user logout (no time-outs)
- Account activation or deactivation
- Password change or reset
- Role change
- System settings management
- Certificate management
- OAuth settings management
- Metadata export
- XML file signatures applied
- Configuration archive export and import
- Identity provider (IdP)/service provider (SP) adapter, IdP token processor, or SP token generator created, modified, or deleted
- IdP/SP default URLs modified
- IdP/SP connection created, modified, or deleted
- Adapter-to-Adapter mapping or token exchange mapping created, modified, or deleted

- Authentication policy contract created, modified, or deleted
- IdP Discovery management
- SP Affiliation created, modified, or deleted
- PingOne for Enterprise account connected, modified, or disconnected
- Session timeout event for the following two scenarios:
    - When an administrator’s session has timed out and they subsequently sign on again, then the session timeout event is retroactively logged.
    - When an administrator’s session is invalidated due to inactivity and a session clean-up is performed by the server’s session management on the administrative console node. The timeout event is logged 10 - 15 minutes after the timeout occurred.

Each entry in the `admin.log` file is on a separate line and represents a single administrator action. The general format of each entry is the same, though specific events are recorded with information relevant to each type. Events are recorded when you click the corresponding **Save** button in the administrative console. Each log entry contains information relating to the event, including:

- The time the event occurred on the PingFederate server
- The username of the administrator performing the action
- The roles assigned to the administrator at the time the event occurred
- The type of event that occurred
- Basic information about the event
- jti (JWT ID)

> **Tip**
>
> The `jti` is the ID of the outbound JSON Web Token (JWT) request. This information is applicable for a `LOGIN_ATTEMPT` event when the PingFederate administrative console authentication scheme is OpenID Connect (OIDC).

- The hash of the inbound access token.

> **Tip**
>
> The hash logging is applicable for a `LOGIN_ATTEMPT` event when the PingFederate administrative console authentication scheme is OIDC. To calculate the hash value for a token or authorization code, run the `calculatehash.sh/bat` script in the PingFederate `bin` folder.

> **Important**
>
> This feature should only be enabled in production environments when actively troubleshooting authentication issues.

Each of these fields is separated by a vertical pipe (`\|`) for easier parsing.

Detailed event logging

You can also configure PingFederate to log additional event information to a separate log file. When you enable detailed event logging, besides writing basic information to `<pf_install>/pingfederate/log/adming.log`, PingFederate logs detailed information about each event to `admin-event-detail.log` in the same log directory.

> **Important**
>
> Events recorded in the log are limited to changes stored in XML files. For example, the log does not record changes to OAuth clients stored in external datastores, such as LDAP directories or Java Database Connectivity (JDBC) databases. Additionally, not all events have detailed information. For instance, sign on attempts are only logged to the `admin.log` file.

PingFederate links events between `admin.log` and `admin-event-detail.log` by a unique event ID. Each entry in the `admin-event-detail.log` file contains:

- The ID of the event
- The name of the file involved
- The type of event that occurred
- The line number where the change occurred
- The changes made

To enable detail event logging, set the `pf.log.eventdetail` property to `true` in the `<pf_install>/pingfederate/bin/run.properties` file.

API audit logging

PingFederate provides API endpoints and management services on the administrative port (9999) and the runtime port (9031) that are logged for auditing purposes.

Actions performed through these endpoints are logged for auditing purposes, as described in the following table.

API	Port	Log File
Administrative API	Administrative Port	`admin-api.log`
OAuth Client Management Service	Runtime Port	`runtime-api.log`
OAuth Access Grant Management Service	Runtime Port	`runtime-api.log`
Session Revocation API	Runtime Port	`runtime-api.log`

Administrative API audit log

PingFederate records actions performed through the administrative API in the `<pf_install>/pingfederate/log/admin-api.log` file.

While the events are not configurable, Log4j 2 configuration settings in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file can be adjusted to deliver the desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method
- REST endpoint
- HTTP status code
- jti (JWT ID)

> **Tip**
>
> The `jti` is the ID of the outbound JSON Web Token (JWT) request. This information is applicable when the PingFederate administrative API authentication scheme is OAuth2 and the client authentication method is *private_key_jwt*.

- The hash of the inbound access token

> **Tip**
>
> The hash logging is applicable when the PingFederate administrative API authentication scheme is OAuth2. To calculate the hash value for a token or authorization code, run the `calculatehash.sh/bat` script in the PingFederate `bin` folder.

> **Important**
>
> This feature should only be enabled in production environments when actively troubleshooting authentication issues.

- HTTP request header
- TLS version

> **Note**
>
> The `<pf_install>/pingfederate/log/admin-api.log` does not include the HTTP request header and TLS version values by default. You can customize this log to include additional or less information by modifying the pattern elements in the `log4j2.xml` file. For more information, see Log4j 2 logging service and configuration.

Each of these fields is separated by a vertical pipe (`\|`) for ease of parsing.

> **Note**
>
> PingFederate also records actions performed through the administrative API in the `<pf_install>/pingfederate/log/admin.log` file. For more information, see Administrator audit logging.

Runtime APIs audit log

PingFederate records actions performed through the OAuth Client Management Service, the OAuth Access Grant Management Service, and the Session Revocation API in the `<pf_install>/pingfederate/log/runtime-api.log` file.

While the events are not configurable, Log4j 2 configuration settings in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file can be adjusted to deliver the desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method
- REST endpoint
- HTTP status code
- HTTP request header
- TLS version

> **Note**
>
> The `<pf_install>/pingfederate/log/runtime-api.log` does not include the HTTP request header and TLS version values by default. You can customize this log to include additional or less information by modifying the pattern elements in the `log4j2.xml` file. For more information, see Log4j 2 logging service and configuration.

Each of these fields is separated by a vertical pipe (`\|`) for ease of parsing.

Runtime transaction logging

PingFederate provides for flexible, scalable logging of all federated-identity transactions, for both inbound and outbound messages.

About this task

Administrators can configure transaction logging to any of the four modes on a per-connection basis or override the logging mode for all service provider (SP) connections, identity provider (IdP) connections, or both for troubleshooting or as a one-step means of raising or lowering all connection logging modes to the same level. The log file is `transaction.log`, located in the `<pf_install>>/pingfederate/log` directory.

The following table describes the four transaction logging modes.

Mode	Description
No Logging	No transaction logging.
Standard	(Default) Summary information for each transaction message, including: • Time stamp • Hostname and port • Log mode • Connection ID • SAML status code for SAML responses only • Context • Message type • SAML ID for SAML messages only • Endpoint for outbound messages only • Target URL if single sign-on (SSO) transaction
Enhanced	Includes everything logged at the **Standard level** including: • SAML_SUBJECT* • Binding • Relay state, if available • Signature policy • Signature status • HTTP request parameters, for outbound messages only • Only when available in a SAML assertion, a single logout (SLO) request, an STS Request Security Token Response (RSTR), or an authentication request (AuthnRequest)
Full	Includes everything logged at the **Enhanced level** plus the complete XML message for every transaction.

Each field is separated by a vertical pipe (`\|`) for parsing.

Steps

- To configure transaction logging mode on a per connection basis:
  1. Select the applicable connection on the **IdP Connections** page (**Authentication > Integration > IdP Connections**) or the **SP Connections** page (**Applications > Integration > SP Connections**).
  2. On the **General Info** tab, select one of the logging modes.

- To override transaction logging mode for all IdP or SP connections:

  1. Go to **System > Server > General Settings**.
  2. For IdP connections, select a logging mode in the **IDP Connection Transaction Logging Override** list.
  3. For SP connections, select a logging mode in the **SP Connection Transaction Logging Override** list.
  4. Click **Save**.

Security audit logging

PingFederate records a subset of transaction log information with additional details at runtime, intended to facilitate security auditing and regulatory compliance.

The system records activities from single sign-on (SSO), single logout (SLO), OAuth, WS-Trust Security Token Service (STS), and System for Cross-domain Identity Management (SCIM) inbound provisioning transactions in the security audit log, the `audit.log` file, located in the `<pf_install>/pingfederate/log` directory. You can output security audit log information to different formats, including databases, CEF, and Splunk.

> **Note**
>
> Outbound provisioning transactions are not included in the security audit log. Instead, they are recorded in the outbound provisioning audit log, the `provisioner-audit.log` file, located in the `<pf_install>/pingfederate/log` directory.

The following tables describe the default and available fields. PingFederate separates each field by a vertical pipe (`\|`). As needed, fields are configurable by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

Default fields, in the order that PingFederate records them in the security audit log

Field	Description
%d	The transaction time.
trackingid	The tracking ID values uniquely identify user sessions, useful for correlating log messages in the audit and server logs.
transactionid	An identifier unique to the SSO or SLO PingFederate transaction. **Tip** The unique transaction ID is exposed at the SDK level to help custom plugin implementation simplify transaction tracking across multiple products and services.

Field	Description
event	The type of transaction. For example, `SSO` , `OAuth` , `AUTHN_ATTEMPT` , `AUTHN_REQUEST` , `AUTHN_SESSION_CREATED` , `AUTHN_SESSION_USED` , `AUTHN_SESSION_DELETED` , and `SRI_REVOKED` . `AUTHN_ATTEMPT` and `AUTHN_REQUEST` indicate an authentication attempt against an identity provider (IdP) adapter instance and an authentication request sent to another IdP partner, through an IdP connection, respectively. `AUTHN_SESSION_CREATED` and `AUTHN_SESSION_USED` indicate the creation and employing of a PingFederate session, respectively. `AUTHN_SESSION_DELETED` indicates that a PingFederate session has been removed as a result of a front-channel browser-based logout request via the SAML 2.0 or WS-Federation protocol. `SRI_REVOKED` indicates that a PingFederate session has been added to the session revocation list. `USER_KEY_AND_SRI_ASSOCIATED` indicates a unique user key is associated with a PingFederate session.
subject	The subject of the transaction or authentication attempt.
ip	The incoming IP address.
app	The target service provider (SP) application, the email verification endpoint, or the profile management page, when applicable and available.
connectionid	The partner identifier associated with the transaction. The OAuth client ID value for OAuth transactions. The ID of the authentication policy contract referenced by the local identity profile that has been invoked for the purpose of accessing the email verification endpoint or the profile management page.
protocol	The associated identity protocol; for example, `SAML20` or `OAuth20` .
host	The host name or IP address of the PingFederate server.
role	The role PingFederate played for the transaction.
status	The status of the transactions.
adapterid	The ID of an adapter instance. Consider adding the `authenticationsourceid` and `targetsessionid` fields to record additional information about the request.

Field	Description

description	The description of an authentication failure, when such information is available from the authentication source, or an authorization failure from an erroneous OAuth authorization request.
	For user sign-on events processed through an HTML form adapter, the description includes the error source and error message.
	Here's an example of the description when authentication with an HTML form adapter fails because a data store locked the user's account, where `source` specifies the system ID of the data store:
	`description=[source:LDAP-8C4A5F60684C90B9ECE88D2B] Account Locked`
	Here's an example of the description when authentication with an HTML form adapter fails because PingFederate's Account Locking Service locked the user's account:
	`description=[source:AccountLockingService] Account Locked`

responsetime	The time elapsed in milliseconds from when the system receives a final request for a transaction, to when the system writes the audit message. This value serves as an approximation of total transaction processing time and can be useful for monitoring trends.

Other available fields, in alphabetical order

Field	Description
accessgrantguid	The GUID of the OAuth access grant, for OAuth transactions.
assertionid	The unique ID for the SAML assertion.
atjti	The `jti` (JWT ID) of the outbound JSON Web Token (JWT) access token (when `role` = *AS*).
attrackingid	The tracking ID for OAuth access token. You can use this ID to analyze the flow of OAuth access tokens in the audit log and between PingFederate and PingAccess.
attributes	The user attributes received (for an SP log), sent (for an IdP log), or provided by the user through the self-service registration or profile management page.
authenticationsourceid	An array of one or more IdP adapters, one or more IdP connections, and identity profile, if any, invoked in an authentication or logout flow. For example, `[adapter.HTMLFormSimplePCV, idpConnection.IdP, localIdentity.A8me9rySDn1aIM48]`
authnsessionexpiry	The expiry of an authentication session that has just been created or used.
connectionname	The partner name associated with the transaction. The OAuth client name for OAuth transactions. The name of the authentication policy contract referenced by the local identity profile that has been invoked for the purpose of accessing the email verification endpoint or the profile management page.

Field	Description
granttype	The OAuth grant type.
header\ *{anHttpRequestHeader}*	The HTTP request header value identified by the header name. The header name is case-insensitive. For example, `header{user-agent}` and `header{User-Agent}` are equivalent. To record multiple headers, repeat the `header` field, as illustrated in the following sample pattern. `<pattern>...| %header\{accept-language}| %header\{dnt}  %n</pattern>` Given this partial sample, PingFederate includes both the `accept-language` and `dnt` HTTP request header values when recording entries in the audit log. **Note** To record values from all HTTP request headers, look for the `org.sourceid.servlet.filter.HttpRequestHeaderFilter` Logger in the `log4j2.xml` file. This capability is turned off by default and is likely suitable only for testing and troubleshooting purposes.
httprequestid	The ID of the HTTP request. This can be used for correlation across external systems (like PingDirectory) and for debugging purposes in the server log. This field is optional.
idjti	The `jti` of the outbound ID token (when `role` = *AS*) or inbound ID token (when `role` = *SP*).
inachash	The hash of the inbound authorization code (when `role` = *AS*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
inathash	The hash of the inbound access token (when `role` = *AS* or when `role` = *SP*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
initiator	The federation role that initiated the SSO or SLO: `SP` or `IDP`. Applicable only to SAML 2.0 transactions.
inmessagetype	The incoming message type. Possible values are `Request` or `Response`.
inresponseto	The value of the `InResponseTo` attribute of an SSO or SLO response.
inxmlmsg	The incoming message. For example, a SAML AuthnRequest or the information pertaining to an OAuth request.
inrthash	The hash of the inbound refresh token (when `role` = *AS*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
localuserid	The local ID used for the transaction, when account linking is enabled at the SP.

Field	Description
outachash	The hash of the outbound authorization code (when `role` = *AS*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
outathash	The hash of the outbound access token (when `role` = *AS*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
outrthash	The hash of the outbound refresh token (when `role` = *AS*). NOTE: This feature should only be enabled in production environments when actively troubleshooting authentication issues.
outurl	The URL where the protocol response was sent. For security reason, parameters and fragments are excluded.
outxmlmsg	The outgoing message. For example, a SAML Response or the information pertaining to a response for an OAuth request.
parameter\\{*anHttpRequestParameter*}	The value of the HTTP request parameter identified by the parameter name. The parameter name is case-sensitive. To record multiple parameters, repeat the `parameter` field, as illustrated in the following sample pattern. `<pattern>...| %parameter\{foo1}| %parameter\{Foo3}  %n</pattern>` Given this partial sample, PingFederate includes both the `foo1` and `Foo3` HTTP request parameter values when recording entries in the audit log. **Note** To record values from all HTTP request parameter, look for the `org.sourceid.servlet.filter.HttpRequestParameterFilter` Logger in the `log4j2.xml` file. This capability is turned off by default and is likely suitable only for testing and troubleshooting purposes.
pfversion	The PingFederate version.
requestid	The ID of a SAML request.
requestjti	The `jti` (s) of the inbound JWT access token (when `role` = *AS*), inbound JWT request object (when `role` = *AS*), or outbound JWT request object (when `role` = *SP*).
requeststarttime	The start time of the request in milliseconds since midnight, January 1, 1970 UTC.
responseid	The ID of a SAML response.
sessiongroupid	The internal ID for a group of persistent authentication sessions associated with a single browser instance through the PF.PERSISTENT cookie. It is only set if the request has triggered a session lookup.

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>sri</td><td>The session reference identifier (SRI) for the user, which can be passed to the session revocation API to revoke the user's sessions. It is only set if the request has triggered a session lookup.</td></tr>
<tr><td>stspluginid</td><td>The ID for the token processor or token generator instance.
Applicable only to WS-Trust STS transactions.</td></tr>
<tr><td>targetsessionid</td><td>An array of one or more SP adapters or SP connections invoked in an authentication or logout flow.</td></tr>
<tr><td>tlsversion</td><td>The connection's TLS version.
ⓘ **Note**
If you deploy PingFederate behind a reverse proxy that terminates TLS connections, you can configure the proxy to forward the TLS version in a header. For more information, see the header \{*anHttpRequestHeader*} field and description.</td></tr>
<tr><td>trackedparameter\{`anHttpRequestParameter`}</td><td>The value of the tracked HTTP request parameter identified by the parameter name. The parameter name is case-sensitive.
Tip
The PingFederate policy engine is capable of tracking HTTP request parameters that it receives from the initial request and making them available to authentication sources, selector instances, and contract mappings throughout the policy. As needed, parameters can be configured as such on the **Tracked HTTP Parameters**window. For more information about tracked parameters, see Policies.
To record multiple parameters, repeat the `trackedparameter` field, as illustrated in the following sample pattern.
`<pattern>...| %trackedparameter\{foo2}| %trackedparameter\{Foo4}  %n</pattern>`
Given this partial sample, PingFederate includes both the `foo2` and `Foo4` HTTP request parameter values when recording entries in the audit log.
If the parameter, as indicated by <*anHttpRequestParameter*>, has not been configured as a parameter to be tracked by the policy engine, PingFederate does not record the parameter value in the audit log.</td></tr>
<tr><td>uniqueuserkey</td><td>The unique user key tied to the user's authentication sessions. It is only set if the user authenticated using an IdP adapter that has configured a unique user key attribute.</td></tr>
<tr><td>validatorid</td><td>The ID of the Password Credential Validator (PCV) instance, for the successful attempts.</td></tr>
<tr><td>virtualserverid</td><td>The virtual server ID of a request, if applicable.</td></tr>
</table>

Tip

To calculate the hash value for a token or authorization code, run the `calculatehash.sh/bat` script in the PingFederate `bin` folder.

Related links

- Logging in other formats
- Outbound provisioning audit logging

Outbound provisioning audit logging

The PingFederate `provisioner-audit.log` file records outbound provisioning transactions, intended to facilitate security auditing.

The `provisioner-audit.log` log file is located in the `<pf_install>/pingfederate/log` directory. Outbound provisioning audit log information can be output to different formats, including database and Splunk.

The following table describes all recorded elements. Optionally, you can configure elements by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

Item	Description
%d	Transaction time.
cycle_id	The unique ID for each provisioning cycle.
channel_id	The unique ID of the provisioning channel between source and target.
event_type	The type of provisioning events, such as CREATE and UPDATE.
source_id	The provisioning Source ID.
target_id	The provisioning Target ID.
is_success	A flag to show whether the event was successful or not. If the attempt succeeded, the value is `true`; otherwise, the value is `false`.
non_success_ cause	Description of failure cause.

Related links

- Logging in other formats

Server logging

When PingFederate is configured to log `DEBUG` messages for troubleshooting purposes, it records all runtime and administrative events that can be used for troubleshooting in the `<pf_install>/pingfederate/log/server.log` file, including status and error messages.

Server log information can be output to a database server.

> **Note**
>
> `DEBUG` messages are turned off by default. For troubleshooting purpose, you can re-enable it by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

The following table describes the recorded elements. Optionally, you can configure elements by editing the `log4j2.xml` file as well.

Item	Description
%d	Event date and time.
%X{trackingid}	The tracking ID values uniquely identify user sessions, useful for correlating log messages in the audit and server logs.
%p	Logging level.
%c	The Java class issuing the status or error message, when applicable.
%m	Status or error message.

To facilitate troubleshooting, administrators can use a filter utility to aggregate related events using the log filter tool.

Related links

- Writing logs to databases
- Enabling console logging

Server log filter

PingFederate provides a utility, **`logfilter`**, that administrators can use to filter server logs.

The **`logfilter`** utility is located in the `<pf_install>/pingfederate/bin` directory, `logfilter.bat` for Windows, and `logfilter.sh` for Linux.

The utility sorts through all the server logs in the log directory. Administrators can move or copy one or more server log files to a different directory that can be specified as an input parameter.

The log filter returns lists of log entries based on either:

- Entity ID and subject

- Tracking ID
- Session cross-reference ID

The following table describes the utility's command options. The table afterward describes optional parameters available for all of the commands.

Server log filter command parameters

Command parameter	Description
-entityid `entity ID` -subject `subject`	These two commands must be used together and return a list of transactions for the specified federation partner's entity ID and transaction subject.
-trackingid `tracking ID`	This command returns a list of transactions with the same tracking ID.
-sessionxrefid `session cross-reference ID`	This command returns a list of transactions for an ID assigned by PingFederate to associate different transactions according to the user session under which they occurred. The value of `session cross-reference ID` can be the value of any of the following transaction tags in the target server logs: • Artifact • Session Index • Assertion ID

Server log filter parameters (optional)

Parameter	Description
-logsdir `log files directory`	Full or relative path to source directory for the logs. Default: all `server.log` files are written to the `<pf_install>/pingfederate/log` directory, a setting that can be adjusted by the `pf.log.dir` property in the `<pf_install>/pingfederate/bin/run.properties` file.
-outputfile `output file`	Output path and file for the returned list. Default: `$<pf.log.dir>/logfilter_output.log`.
-outputtoconsole	Returns list to the command console rather than to a file.

The log filter creates its own log file, `logfilter.log`, located in the log directory. Optionally, administrators can control settings for this log in the `<pf_install>/pingfederate/bin/logfilter.log4j2.xml` file.

Logging in other formats

PingFederate provides the option of writing the audit log, the provisioner audit log, the provisioner log, and the server log to commonly used databases with failover to file logging.

For the audit log and the provisioner audit log, administrators can choose instead to write the information to the Common Event Format (CEF), a differently formatted log file that can be used by Splunk, or both.

Writing logs to databases

Database logging replaces file logging. For each qualified database server, PingFederate provides scripts to create database tables for the audit log, the provisioner audit log, the provisioner log, and the server log.

About this task

You can find these scripts in the `<pf_install>/pingfederate/server/default/conf/log4j/sql-scripts` directory.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers.

Failover file logging is provided in the event that database logging fails for any reasons. By default, PingFederate retries database logging every minute. Messages written to log files during failover periods are not copied over to the database server.

You enable database logging for the audit log, the provisioner audit log, the provisioner log, and the server log in the `log4j2.xml` file.

Steps

1. Edit `<pf_install>/pingfederate/server/default/conf/log4j2.xml`.
2. After the `Preserve messages in a local file` section, for each log that you want to enable database logging, uncomment the preset Java Database Connectivity (`JDBC`) appender configuration based on the choice of your database server.

   ### *Audit log*

   - Oracle MySQL - `SecurityAuditToMySQLDB`
   - Oracle Database - `SecurityAuditToOracleDB`
   - PostgreSQL - `SecurityAuditToPostgreSQLDB`
   - Microsoft SQL Server - `SecurityAuditToSQLServerDB`

   ### *Provisioner audit log*

   - Oracle MySQL - `OutboundProvisionerEventToMySQLDB`
   - Oracle Database - `OutboundProvisionerEventToOracleDB`
   - PostgreSQL - `OutboundProvisionerEventToPostgreSQLDB`
   - Microsoft SQL Server - `OutboundProvisionerEventToSQLServerDB`

   ### *Provisioner log*

   - Oracle MySQL - `ProvisionerLogToMySQLDB`

- Oracle Database - `ProvisionerLogToOracleDB`
- PostgreSQL - `ProvisionerLogToPostgreSQLDB`
- Microsoft SQL Server - `ProvisionerLogToSQLServerDB`

Server log

- Oracle MySQL - `ServerLogToMySQLDB`
- Oracle Database - `ServerLogToOracleDB`
- PostgreSQL - `ServerLogToPostgreSQLDB`
- Microsoft SQL Server - `ServerLogToSQLServerDB`

> **Note**
>
> Each `JDBC` appender is followed by two related appenders, `PingFailover` and `RollingFile`. Together, they create a running `*-failover.log` file in the log directory in the event that database logging fails for any reason. Both appenders must also be enabled (uncommented).

> **Tip**
>
> For more information about each appender, review inline comments and notes in the `log4j2.xml` file.

3. Replace placeholder parameter values in `log4j2.db.properties` in the same `conf` directory for the applicable Java Database Connectivity (JDBC) servers.

   The parameter values provide access to the database. Test and validate access prior to production deployment. Like `log4j2.xml`, `log4j2.db.properties` is also individually managed per PingFederate server. This flexibility allows multiple PingFederate nodes in a clustered environment to write messages to different destinations, as needed.

   > **Tip**
   >
   > You can obfuscate the password used to access the database by running the **`obfuscate`** utility, located in the `<pf_install>/pingfederate/bin` directory: `obfuscate.bat` for Windows or `obfuscate.sh` for Linux. Use the actual password as an argument and copy the entire result into the value for the password parameter in `log4j2.db.properties`.

4. Uncomment the appender reference, `<AppenderRef/>`, in the associated logger elements, as described inline in the `log4j2.xml` file.

Audit log

Uncomment the corresponding `PingFailover` appender references from the following `Logger` elements located under the `Loggers` section:

- Browser SSO SP and adapter-to-adapter - `org.sourceid.websso.profiles.sp.SpAuditLogger`
- Browser SSO IdP and adapter-to-adapter - `org.sourceid.websso.profiles.idp.IdpAuditLogger`
- OAuth authorization server - `org.sourceid.websso.profiles.idp.AsAuditLogger`

- ◦ Dynamic Client Registration - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger`
- ◦ WS-Trust STS, IdP, and SP - `org.sourceid.wstrust.log.STSAuditLogger`

Provisioner audit log

Uncomment the corresponding `PingFailover` appender reference from the `ProvisionerAuditLogger` `Logger` element located under the `Set up the Outbound provisioner audit logger` section.

Provisioner log

Uncomment the corresponding `PingFailover` appender reference from the `com.pingidentity.provisioner` `AsyncLogger` element located under the `Loggers` section.

Server log

Uncomment the corresponding `PingFailover` appender reference from the `root` element located under the `Set up the Root Logger` section, near the end of the file.

> **Important**
>
> As indicated in the IMPORTANT comments for the loggers, you must also remove some of the existing appender references.

5. **Optional:** For the audit log and the provisioner audit log, you can configure elements for database logging in the `ConversionPattern` appender parameter, as needed.

Logging in Common Event Format

You can use PingFederate to write from logs using the Common Event Format (CEF) open standard.

CEF is an open logging standard. PingFederate provides an option of writing elements from the audit log, the provisioner audit log, or both at runtime to a syslog receiver for parsing and analysis using ArcSight from Micro Focus. Alternatively, administrators can write the information to a flat file in CEF. However, you should use syslog when available.

> **Note**
>
> PingFederate is tested with ArcSight for interoperability using the default elements defined in `log4j2.xml`. Any additions to these elements might render your CEF logging incompatible with ArcSight.

Writing audit log in CEF

You can write the audit log in Common Event Format (CEF) in PingFederate.

Steps

1. Edit `<pf_install>/pingfederate/server/default/conf/log4j2.xml`.

2. Under the `Security Audit log : CEF Formatted syslog appender` section, uncomment one of the preset appender configurations:

   - `SecurityAuditToCEFSyslog` - a `Socket` appender
   - `SecurityAuditToCEFFile` - a `RollingFile` appender

   > **Note**
   >
   > The `SecurityAuditToCEFSyslog` `Socket` appender is followed by two related appenders, `PingFailover` and `RollingFile`. Together, they create a running `audit-cef-syslog-failover.log` file in the log directory in the event that CEF logging fails for any reason. Both appenders must also be enabled and uncommented.

   > **Tip**
   >
   > Review inline comments and notes in the `log4j2.xml` file for more information about each appender.

3. If you are configuring the `SecurityAuditToCEFSyslog` `Socket` appender, replace the placeholder parameter values for the syslog host.

4. If you are configuring the `SecurityAuditToCEFSyslog` `Socket` appender. uncomment the `PingFailover` appender reference (`<appender-ref ref="SecurityAuditToCEFSyslog-FAILOVER"/>`) from the following `Logger` elements located under the `Loggers` section:

   - Browser SSO SP and adapter-to-adapter - `org.sourceid.websso.profiles.sp.SpAuditLogger`
   - Browser SSO IdP and adapter-to-adapter - `org.sourceid.websso.profiles.idp.IdpAuditLogger`
   - OAuth authorization server - `org.sourceid.websso.profiles.idp.AsAuditLogger`
   - Dynamic Client Registration - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger`
   - WS-Trust STS, identity provider (IdP), and service provider (SP) - `org.sourceid.wstrust.log.STSAuditLogger`

   > **Important**
   >
   > As indicated in the IMPORTANT comments for the loggers, you must also remove some of the existing appender references.

Writing provisioner audit log in CEF

You can write provisioner audit logs in Common Event Format (CEF) for PingFederate. PingFederate provides an option of writing elements from the audit log and the provisioner audit log at runtime to a syslog receiver for parsing and analysis using ArcSight from Micro Focus.

Steps

1. Edit `<pf_install>/pingfederate/server/default/conf/log4j2.xml`.

2. Uncomment one of the preset appender configurations:

   - `OutboundProvisionerEventToCEFSyslog` (a `Socket` appender under the `Outbound provisioner audit log : CEF Formatted syslog appender` section)

     > **Note**
     >
     > This `Socket` appender is followed by two related appenders, `PingFailover` and `RollingFile`. Together, they create a running `provisioner-audit-cef-syslog-failover.log` file in the log directory in the event that CEF logging fails for any reason. Both appenders must also be enabled (uncommented).

   - `OutboundProvisionerEventToCEFFile` (a `RollingFile` appender under the `Outbound provisioner audit log for CEFFile` section)

     > **Tip**
     >
     > Review inline comments and notes in the `log4j2.xml` file for more information about each appender.

3. If you are configuring the `OutboundProvisionerEventToCEFSyslog` `Socket` appender, replace the placeholder parameter values for the syslog host.

4. If you are configuring the `OutboundProvisionerEventToCEFSyslog` `Socket` appender, uncomment the `PingFailover` appender reference (`<appender-ref ref="OutboundProvisionerEventToCEFSyslog-FAILOVER"/>`) from the `ProvisionerAuditLogger` `Logger` elements located under the `Set up the Outbound provisioner audit logger` section.

   > **Important**
   >
   > As indicated in the IMPORTANT comments for the loggers, you must also remove some of the existing appender references.

Writing audit logs for Splunk

Ping Identity provides a custom Splunk App for PingFederate to process audit logs generated by a PingFederate deployment. Splunk is an enterprise software that allows for monitoring, reporting, and analysis of consolidated log files.

Before you begin

- Download and install Splunk

About this task

Splunk captures and indexes real-time data into a single searchable repository where reports, graphs, and other data visualization can be generated.

The PingFederate Splunk App provides rich system monitoring and reporting, including:

- Current transaction and system reports
- Service reports, such as a daily usage report, and identity provider (IdP) and service provider (SP) reports per connection
- Trend reports, such as weekly and monthly usage reports, and trend analysis

Splunk uses a specially formatted version of the audit log `splunk-audit.log`, which you can write to the PingFederate log directory when you complete the setup steps.

> **Note**
>
> The Splunk App for PingFederate is available separately. It requires enterprise-licensed, or trial installation of the Splunk software and the Splunk Universal Forwarder, which is needed to collect data from the PingFederate audit log for Splunk. The application includes additional documentation on installation and available features. To download the free application, go to splunkbase.splunk.com and search for PingFederate.

Steps

1. Set up your Splunk server.
    1. Enable a receiver to listen for data from the PingFedrate server.

       For more information, see the Splunk documentation.
    2. Install Splunk App for PingFederate.
2. Configure PingFederate to write audit log messages to the `<pf_install>/pingfederate/log/splunk-audit.log` file.
    1. Edit `<pf_install>/pingfederate/server/default/conf/log4j2.xml`.
    2. Locate the following `Logger` elements located under the `Loggers` section:
        - Browser single sign-on (SSO) SP and adapter-to-adapter - `org.sourceid.websso.profiles.sp.SpAuditLogger`
        - Browser SSO IdP and adapter-to-adapter - `org.sourceid.websso.profiles.idp.IdpAuditLogger`
        - OAuth authorization server - `org.sourceid.websso.profiles.idp.AsAuditLogger`
        - Dynamic Client Registration - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger`
        - WS-Trust STS, IdP, and SP - `org.sourceid.wstrust.log.STSAuditLogger`
        - Provisioner Audit Logger - `ProvisionerAuditLogger`
    3. Uncomment the `SecurityAudit2Splunk` `RollingFile` appender reference, `<appender-ref ref="SecurityAudit2Splunk"/>`, from one or more of the `Logger` elements.

       Example:

       For example, the default logger for an IdP audit log reads as follows.

```
<Logger name="org.sourceid.websso.profiles.idp.IdpAuditLogger"
        level="INFO" additivity="false" includeLocation="false">
    <appender-ref ref="SecurityAudit2File" />
    <!--
        <appender-ref ref="SecurityAuditToCEFSyslog-FAILOVER"/>
        <appender-ref ref="SecurityAuditToCEFFile"/>
        <appender-ref ref="SecurityAuditToMySQLDB-FAILOVER"/>
        <appender-ref ref="SecurityAuditToPostgreSQLDB-FAILOVER" />
        <appender-ref ref="SecurityAuditToSQLServerDB-FAILOVER"/>
        <appender-ref ref="SecurityAuditToOracleDB-FAILOVER"/>
        <appender-ref ref="SecurityAudit2Splunk"/>
    -->
</Logger>
```

To log Browser SSO IdP audit log messages to `splunk-audit.log`, update the `Logger` element as follows.

```
<Logger name="org.sourceid.websso.profiles.idp.IdpAuditLogger"
        level="INFO" additivity="false" includeLocation="false">
     <appender-ref ref="SecurityAudit2Splunk"/>
    <!--
        <appender-ref ref="SecurityAuditToCEFSyslog-FAILOVER"/>
        <appender-ref ref="SecurityAuditToCEFFile"/>
        <appender-ref ref="SecurityAuditToMySQLDB-FAILOVER"/>
        <appender-ref ref="SecurityAuditToPostgreSQLDB-FAILOVER" />
        <appender-ref ref="SecurityAuditToSQLServerDB-FAILOVER"/>
        <appender-ref ref="SecurityAuditToOracleDB-FAILOVER"/>
        <appender-ref ref="SecurityAudit2Splunk"/>
         <appender-ref ref="SecurityAudit2File" />
    -->
</Logger>
```

> **Note**
>
> For auditing of adapter-to-adapter events, you must enable both the IdP and SP loggers.

4. Uncomment the following section:

```
<RollingFile name="SecurityAudit2Splunk" fileName="${sys:pf.log.dir}/splunk-audit.log"
filePattern="${sys:pf.log.dir}/splunk-audit.%d

{yyyy-MM-dd}
.log"
ignoreExceptions="false">
<PatternLayout>
<pattern>%d trackingid="%X

{trackingid}
" event=%X

{event}
subject="%X

{subject}
" ip=%X

{ip}
app=%X

{app}
connectionid=%X

{connectionid}
protocol="%X

{protocol}
" pfhost=%X

{host}
role=%X

{role}
status=%X

{status}
adapterid=%X

{adapterid}
description="%X

{description}
" responsetime=%X

{responsetime}
inmessagetype="%X

{inmessagetype}
" %n</pattern>
```

```
</PatternLayout>
<Policies>
<TimeBasedTriggeringPolicy />
</Policies>
</RollingFile>
```

3. Set up Splunk Universal Forwarder.

    1. Download the Splunk Universal Forwarder from Splunk and install it on the PingFederate server.

    2. Configure the Splunk Universal Forwarder to monitor the `splunk-audit.log` file and forward the data to the receiver configured in step 1.

    For detailed installation and configuration instructions, see the Splunk Universal Forwarder documentation.

Alternative console authentication

As an alternative to using PingFederate’s own internal datastore for authentication to the administrative console, you can configure PingFederate to use your network’s LDAP user-datastore, the RADIUS protocol, client certificates, or OIDC-based authentication.

You can configure any of these alternative console authentication methods at any time. Most user-management functions are handled outside the scope of the PingFederate administrative console when alternative authentication is enabled.

Unlike native authentication, for which you configure local accounts and their privileges in **System > Server > Administrative Accounts** , you must define roles in configuration files when using an alternative authentication scheme. Similar to native authentication, PingFederate provides two account types and three administrative roles for role-based access control, as shown in the following table.

PingFederate User Access Control

Account type	Administrative role	Access privileges
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Admin	Admin	Configure partner connections and most system settings, except the management of local accounts and the handling of local keys and certificates.

Account type	Administrative role	Access privileges
Admin	Expression Admin	Map user attributes by using the expression language, Object-Graph Navigation Language (OGNL). **Important** Only Administrative users who have both the Admin role and the Expression Admin role: • Can be granted the User Admin role. This restriction prevents non-Expression Admin users from granting themselves the Expression Admin Role. • Can be granted write access to the file system or directory where PingFederate is installed. This restriction prevents a non-Expression Admin user from placing a `data.zip` file containing expressions into the `<pf_install>/pingfederate/server/default/deploy` directory, which would introduce expressions into PingFederate.
Admin	Crypto Admin	Manage local keys and certificates.
Auditor	Not applicable	View-only permissions for all administrative functions. When the **Auditor** role is assigned, no other administrative roles can be set.

> **Note**
>
> All four administrative roles are required to access and make changes through the following services:
>
> - The `/bulk`, `/configArchive`, and `/configStore` administrative API endpoints
> - The **Configuration Archive** window, accessed from **System > Server**, in the administrative console
> - The **Connection Management** configuration item on the **Service Authentication** window, accessed from **Security > System Integration**

Enabling OIDC-based authentication

You can enable OpenID Connect (OIDC)-based authentication to the administrative console by setting a property in the `run.properties` file, and by configuring other properties in the `oidc.properties` file.

About this task

> **Important**
>
> All endpoints must be HTTPS.

Steps

1. On your OIDC provider, configure an OAuth client to represent the PingFederate administrative console. Specify the following redirect URI for the client:

`https://<pf_admin_hostname>:<pf_admin_port>/pingfederate/app?service=finishsso`

or, if using `pf.admin.baseurl`:

`https://<pf.admin.baseurl>/pingfederate/app?service=finishsso`

You need the client's credentials in the following steps.

2. Edit the `<pf_install>/pingfederate/bin/run.properties` file, and set the `pf.console.authentication` property to `OIDC`.

   Note

   You might need to configure the `pf.admin.baseurl` property as well. This property defines the URL that PingFederate's administrative node uses to populate resource references in Administrative API responses. The administrative node also uses it for the redirect URL it sends to an OpenID Provider for administrator OIDC login (for example, /https://pingfederate-admin.example.com or, if the load balancer uses a custom port, /https://pingfederate-admin.example.com:8443). The default value is blank.
   Use `pf.admin.baseurl` instead of `pf.admin.hostname`. If `run.properties` defines both, PingFederate ignores `pf.admin.hostname`. However, if `run.properties` defines only `pf.admin.hostname`, PingFederate constructs the URL the same way it does in versions of PingFederate before 10.3.

3. Edit the `<pf_install>/pingfederate/bin/oidc.properties` file, and modify the applicable properties as described in the following table.

   Important

   PingFederate begins to validate the properties defined in `oidc.properties` at start up. This will not include validations that PingFederate can only perform during run time, such as validating the value of `issuer` against the value of `iss` from an ID token.
   Incorrectly configured properties in `oidc.properties` can cause PingFederate to fail to start.

4. Restart PingFederate.

`oidc.properties` properties

Property	Description
`client.id`	The client ID to communicate with the OpenID Provider. This property is required.
`client.secret`	The client secret used to communicate with the OpenID Provider. The client secret should be in obfuscated format. We recommend that the secret be obfuscated using one of the following utilities in the `../bin` directory: • On Windows: **`obfuscate.bat`** • On Linux: **`./obfuscate.sh`** For example: `obfuscate.bat secret` This property is required when the client authentication is either client_secret_basic, client_secret_post, or client_secret_jwt.

Property	Description
`client.authn.method`	The type of client authentication that is expected by the token endpoint in the OpenID Provider. Supported values include: • client_secret_basic – Client credentials using the HTTP Basic authentication scheme. • client_secret_post – Client Credentials included in the request body. • private_key_jwt – Client authenticates in accordance with JSON Web Token (JWT). • client_secret_jwt – Client authenticates in accordance with Open ID Connect specification. This property is required.
`authorization.endpoint`	The authorization endpoint at the OpenID Provider. This property is required.
`pushed.authorization.request.endpoint`	The full HTTPS URL of the Pushed Authorization Request (PAR) endpoint at the OpenID Provider. When configured, authorization requests shall be sent to PAR endpoint to obtain the *request_uri*. This property is optional.
`token.endpoint`	The token endpoint at the OpenID Provider. PingFederate OIDC login only supports the authorization code flow. This property is required.
`user.info.endpoint`	The endpoint that is accessed when the required claims are not present in the ID tokens. This property is optional.
`end.session.endpoint`	The end session endpoint at the OpenID Provider. When no value is provided, the administrator will be redirected to the default PingFederate logout page. This property is optional.
`issuer`	The issuer identifier of the OpenID Provider. The value provided is matched with the iss claims in the obtained ID token. This property is required.
`acr.values`	The authentication context class reference values that will be used by the OpenID Provider. This property is optional.
`scopes`	The authorization endpoint at the OpenID Provider. The default setting is `openid`. This property is required.
`username.attribute.name`	The name of the claim that represents the username of the administrator. The default setting is `sub`. This property is required.

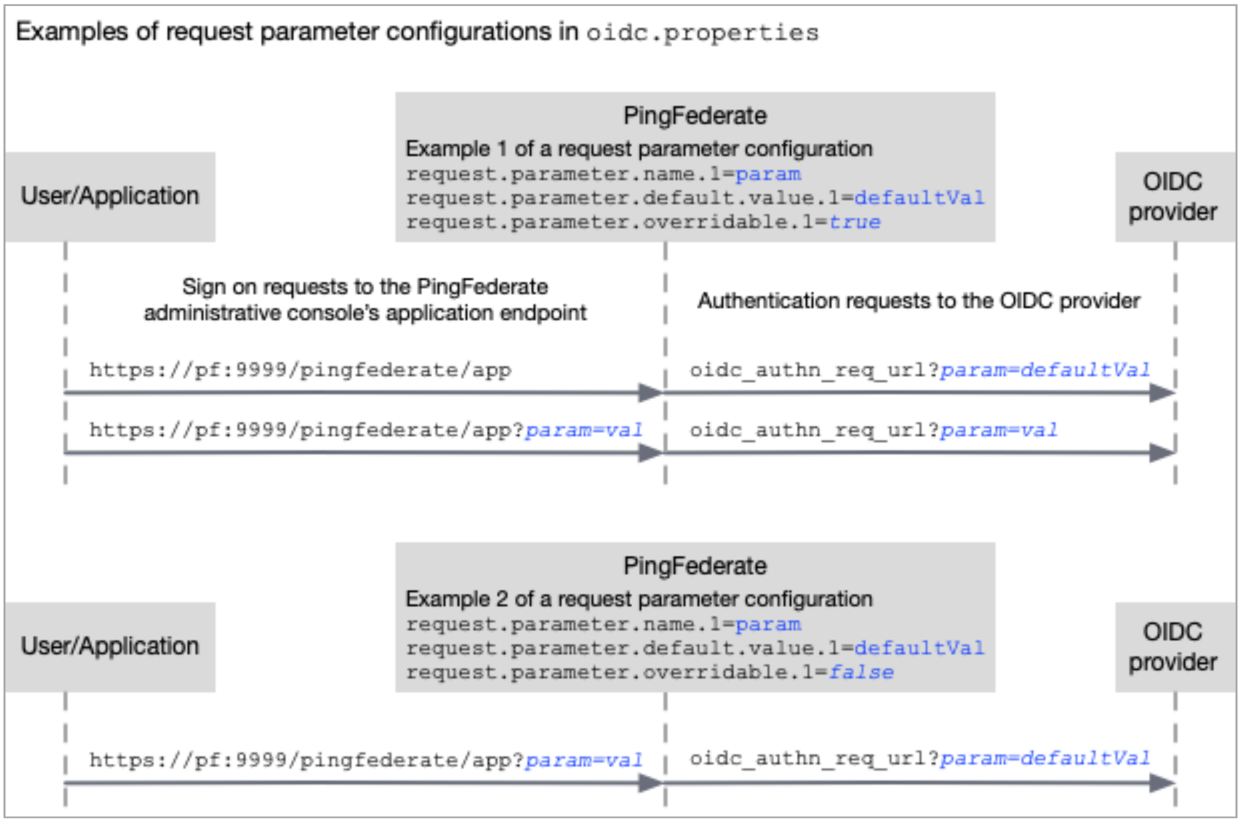
Property	Description
`role.attribute.name`	The name of the claim that is used to determine the role for administrators. This property is required.

`role.map.admin.`*n*	Used when multiple values need to be mapped to a single PingFederate role. In this case, multiple properties must be created using a numeric, incremental suffix, starting with 1. For example:
	`role.map.admin.1=`
	`role.map.admin.2=`
	`role.map.admin.3=`

`role.admin`
`role.cryptoManager`
`role.userAdmin`
`role.expressionAdmin`
	The administrator role claim value mapping. For example, assume that admin_role is a claim and the possible values for it are:
	`role.admin=admin`
	`role.cryptoManager=crypto`
	`role.userAdmin=uadmin`
	`role.expressionAdmin=eadmin`
	When the claim admin_role has more than one value, for example, `admin_role : ["admin","crypto","uadmin"]`, the user will be granted admin, crypto, and user administrator roles.
	When the claim admin_role has a single value, for example, `admin_role: "admin"`, the user will be granted admin role.
	This property is required.

`role.auditor`	The auditor role claim value mapping. This property is optional.
Request parameters	Optional custom properties that you can use to specify allowed incoming parameters and to define static values for outgoing parameters. **Tip** If you're configuring SSO from PingOne to PingFederate, find the request parameters you'll need in Setting up SSO to PingFederate. The diagram below shows two examples of request parameter configurations. In the first configuration, the sign on request can override the default value. However, if the incoming request doesn't provide a value, then the default value is passed to the OIDC provider. In the second configuration, the sign on request cannot override the default value. For more information, see the Request Parameters section of the `oidc.properties` file.

+


Examples of request parameter configurations in oidc.properties
PingFederate
Example 1 of a request parameter configuration
request.parameter.name.1=param
request.parameter.default.value.1=defaultVal
request.parameter.overridable.1=true
User/Application
OIDC provider
Sign on requests to the PingFederate administrative console’s application endpoint
Authentication requests to the OIDC provider
https://pf:9999/pingfederate/app
oidc_authn_req_url?param=defaultVal
https://pf:9999/pingfederate/app?param=val
oidc_authn_req_url?param=val
PingFederate
Example 2 of a request parameter configuration
request.parameter.name.1=param
request.parameter.default.value.1=defaultVal
request.parameter.overridable.1=false
User/Application
OIDC provider
https://pf:9999/pingfederate/app?param=val
oidc_authn_req_url?param=defaultVal

Examples of request parameter configurations in the `oidc.properties` file

Enabling LDAP authentication

You can enable LDAP authentication by using the configuration files located in the `<pf_install>/pingfederate/bin` directory.

About this task

When LDAP authentication is configured, PingFederate does not lock out administrative users based upon the number of failed sign-on attempts. Instead, responsibility for preventing access is delegated to the LDAP server and enforced according to its password lockout settings.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, change the value of the `pf.console.authentication` property as shown. `pf.console.authentication=LDAP`
2. In the `<pf_install>/pingfederate/bin/ldap.properties` file, change property values as needed for your network configuration.

   For more information, see the comments in the file.

   The roles configured in the properties file apply to both the administrative console and the administrative API.

   > **Important**
   >
   > Remember to assign LDAP users or designated LDAP groups to at least one of the PingFederate administrative roles as indicated in the properties file.

> **Tip**
>
> You can also use this configuration file in conjunction with RADIUS authentication to determine permissions dynamically through an LDAP connection.

3. Start or restart PingFederate.

Enabling RADIUS authentication

You can enable RADIUS authentication using the configuration files located in the `<pf_install>/pingfederate/bin` directory. The RADIUS protocol provides a common approach for implementing strong authentication in a client-server configuration.

About this task

PingFederate supports the protocol scenarios for one-step authentication, such as appending a one-time passcode obtained from an authenticator to the password, and two-step authentication, such as through a challenge-response process.

> **Note**
>
> When RADIUS authentication is configured, PingFederate does not lock out administrative users based on the number of failed sign-on attempts. Instead, responsibility for preventing access is delegated to the RADIUS server and enforced according to its password lockout settings.

> **Note**
>
> The NAS-IP-Address attribute is added to all Access-Request packets sent to the RADIUS server. The value is copied from the `pf.engine.bind.address` property in `run.properties`. Only IPv4 addresses are supported.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, change the value of the `pf.console.authentication` property as shown. `pf.console.authentication=RADIUS`
2. In the `<pf_install>/pingfederate/bin/radius.properties` file, change property values as needed for your network configuration.

   For more information, see the comments in the file.

   The roles configured in the properties file apply to both the administrative console and the administrative API.

   > **Important**
   >
   > Be sure to assign RADIUS users or designated RADIUS groups to at least one of the PingFederate administrative roles as indicated in the properties file. Alternatively, you can set the `use.ldap.roles` property to `true` and use the LDAP properties file, also in the `bin` directory, to map LDAP group-based permissions to PingFederate roles.

3. Start or restart PingFederate.

Multi-factor console authentication using PingID

PingID is a cloud service that enables multi-factor authentication (MFA) using a mobile application.

The PingFederate administrative console supports authentication through the RADIUS protocol, which provides a common approach for implementing strong authentication in a client-server configuration.

By combining these two capabilities, you can configure PingID to provide MFA to protect access to the PingFederate administrative console, which meets the requirement of stronger authentication for administrators accessing security-related software products.

> **Note**
>
> PingID requires a separate license. Please contact sales@pingidentity.com or request a trial license at pingidentity.com.

Requirements

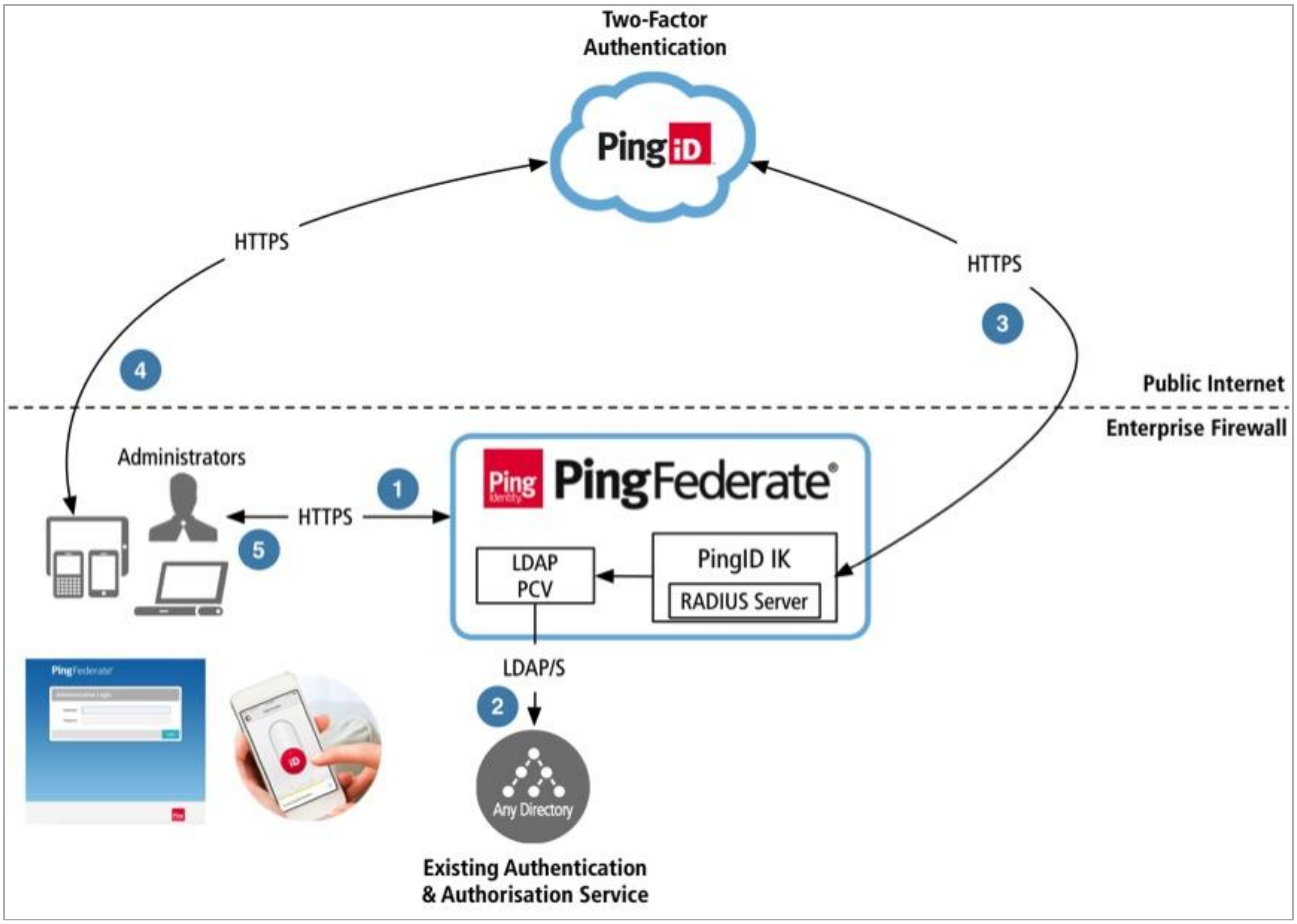
The following components are required to enable MFA to the administrative console using PingID:

- PingFederate with external access to the PingID cloud service
- A PingID license
- A directory server where the administrative credentials and group membership are stored

Solution overview

You can enable multi-factor console authentication using PingID.

When PingID is the second authentication factor for the PingFederate administrative console, the administrators must authenticate successfully against the first factor, such as a directory server, and subsequently respond to the request for authentication from the PingID app on their mobile devices.



Multi-factor Console Authentication using PingID

Processing steps

1. An administrator opens a browser and accesses the PingFederate administration console.
    1. The administrative console displays the Sign On page.
    2. The administrator enters the correct username and password.
2. PingFederate invokes the PingID Password Credential Validator (PCV) to validate the username and password against your directory server.

   The PingID PCV comes with a built-in RADIUS server, which can be used as the point of authentication for the PingFederate administration console using RADIUS authentication.
3. Upon successful validation of the user credentials, the PingID PCV invokes the PingID service with the username.

   The PingID service looks for the username in its datastore.

   If the administrator has not registered a device for use with PingID, the PingID service returns a “username unknown” message. The administrative console displays a device registration window. The administrator must register the mobile device.

4. If the administrator has a registered device, the PingID service notifies the PingID app on the device or sends a text message (SMS) or voice callback message, depending on the configuration for that user account.

    1. The administrator responds to the request for authentication from PingID.

    2. If the administrator has successfully authenticated to the PingID notification, the PingID service returns a “success” message to the PingID PCV.

5. The administrative console menu opens.

Configuring your PingID account

To use PingID as the second factor to authenticate into the PingFederate administrative console, you need a PingID or PingOne account. You must also configure the account to allow the PingID Password Credential Validator (PCV) to use the PingID service.

Steps

1. Register a PingID account.

    1. Contact sales@pingidentity.com for a registration key.

    2. Register at PingOne admin portal.

    3. Select **PingOne for Enterprise** and then click **Next**.

        Let the PingOne admin portal guide you through the registration process. At the end, a confirmation email is sent to your email address.

    4. Open the confirmation email and follow the instructions to activate your PingOne for Enterprise account.

2. Enable PingID client integration.

    1. Sign on to the PingOne admin portal.

    2. Go to **Setup** > **PingID** > **Client Integration**.

    3. Under **Integrate with PingFederate and other Clients**, click **Download** and save the `pingid.properties` file for a subsequent task.

    4. Sign off of the PingOne admin portal.

Creating an LDAP Username Password Credential Validator instance

You can create an LDAP username password credential validator (PCV) in the PingFederate administrative console to create a second factor for multi-factor authentication (MFA).

About this task

Administrators must authenticate successfully against the first factor, such as a directory server where the administrator accounts, credentials and group memberships are stored. To fulfill this requirement, you need an LDAP connection from PingFederate to your directory server, and an instance of the LDAP Username Password Credential Validator.

Steps

1. Go to **System > Data & Credential Stores > Password Credential Validators**. On the **Password Credential Validators** window, click **Create New Instance**.

2. On the **Type** tab, from the **Type** list, select the **LDAP Username Password Credential Validator** and complete the **Instance Name** and **Instance ID** fields.

3. On the **Instance Configuration** tab, from the **LDAP datastore** list, select the datastore and complete the **Search Base** and **Search Filter** fields.

   For more information about each field, see the following table.

Field	Description
LDAP Datastore (Required)	The LDAP datastore configured in PingFederate. If you have not configured the server to communicate with the LDAP directory server you need, click **Manage Data Stores**. There is no default selection.
Search Base (Required)	The location in the directory server where the search begins. This field has no default value.
Search Filter (Required)	The LDAP query to locate a user record. If your use case requires the flexibility of allowing users to identify themselves using different attributes, you can include these attributes in your query. For instance, the following search filter allows users to sign on using either the `sAMAccountName` or `employeeNumber` attribute value through the HTML Form Adapter: `(|(sAMAccountName=${username})(employeeNumber=${username}))` This field has no default value.
Scope of Search	The level of search to be performed in the search base. **One Level** indicates a search of objects immediately subordinate to the base object, not including the base object itself. **Subtree** indicates a search of the base object and the entire subtree within the base object distinguished name. The default selection is **Subtree**.
Case-Sensitive Matching	The option to enable case-sensitive matching between the LDAP error messages returned from the directory server and the **Match Expression** values specified on this window. This check box is selected by default.

4. On the **Extended Contract** tab, click **Next** to skip to the **Summary** tab.

5. On the **Summary** tab, review the configuration, modify as needed, and then save the configuration.

Related links

- Datastores
- Configuring the LDAP Username Password Credential Validator

Configuring a PingID Password Credential Validator instance

You can create and configure an instance of the PingID Password Credential Validator in the PingFederate administrative console.

Before you begin

Open the previously downloaded `pingid.properties` file in a text editor, copy its content, and then close the file.

Steps

1. Go to **System > Data & Credential Stores > Password Credential Validators**, and then click **Create New Instance**.
2. On the **Type** tab, from the **Type** list, select **PingID PCV (with integrated RADIUS server)** and complete the **Instance Name** and **Instance ID** fields.
3. On the **Instance Configuration** tab, configure the required fields as follows.
    1. Click **Add a new row to 'RADIUS Clients'**, enter `127.0.0.1` as the RADIUS IP address and a value in the **Client Shared Secret** field. Click **Update**.

    > **Tip**
    >
    > `127.0.0.1` represents the local RADIUS client, the PingFederate administrative console, which calls the RADIUS server bundled in the PingID PCV for authentication.

    The **Client Shared Secret** value is required for the next task.

    1. Click **Add a new row to 'Delegate PCV's'**, select the previously created LDAP Username Credential Validator instance, and then click **Update**.
    2. Paste the content from the `pingid.properties` file into the **PingID Properties File** field.
    3. Review the rest of the default settings. Modify as needed to meet your requirements, and click **Next**.
4. On the **Extended Contract** tab, click **Next** to skip to the **Summary** tab.
5. On the **Summary** tab, review, modify if needed, and save the configuration.

Related links

- RADIUS PCV Parameters Reference Guide

Configuring PingFederate to use RADIUS authentication

You can enable RADIUS authentication in the PingFederate administrative console.

About this task

In this multi-factor console authentication use case, the PingFederate administrative console is a RADIUS client that calls the local RADIUS server bundled in the PingID Password Credential Validator (PCV) for the second factor authentication.

> **Note**
>
> For a clustered PingFederate environment, perform these steps on the console node.

Steps

1. Open the `<pf_install>/pingfederate/bin/run.properties` file in a text editor and set the `pf.console.authentication` property to `RADIUS`. `pf.console.authentication=RADIUS`
2. Obfuscate the **Client Shared Secret** value using a PingFederate command-line tool.

   Example:

   Windows: `<pf_install>\pingfederate\bin\obfuscate.bat clientSharedSecret`

   Linux: `<pf_install>/pingfederate/bin/obfuscate.sh clientSharedSecret`

   Result:

   The output should be a long line of text.
3. Copy the output for the next step.
4. Open the `<pf_install>/pingfederate/bin/radius.properties` file in a text editor and modify as follows.

   ```
   host=<host>
   shared.secret=obfuscatedClientSharedSecret
   timeout=10000
   ```

   > **Tip**
   >
   > For a clustered PingFederate environment, the `host` value must be a runtime engine IP address or a hostname.
   > The `timeout` value is the number of milliseconds to wait for the second authentication factor to complete before timing out the login attempt. In this use case, ten seconds, or `10000` ms, should be sufficient for PingID.

   In addition, assign one or more RADIUS users or designated RADIUS groups to at least one of the PingFederate administrative roles as indicated in the `radius.properties` file. Alternatively, you can set the `use.ldap.roles` property to `true` and use the LDAP properties file, `ldap.properties` in the same `bin` directory, to map LDAP group-based permissions to PingFederate roles.
5. Save your changes, and restart PingFederate.

Related links

- Administrative accounts

Verifying your setup

Verify your setup after you have completed the required configuration.

Steps

1. Start PingFederate.

   In a clustered PingFederate environment, start PingFederate on the console node.

2. Start a web browser.

3. Browse to the URL.

   https://<*pf_host*>:9999/pingfederate/app where <*pf_host*> is the network address of your PingFederate server. It can be an IP address, a host name, or a fully qualified domain name. It must be accessible from your computer.

4. Authenticate using your directory user credentials.

   Upon successful validation, the PingFederate administrative console prompts you to authenticate using PingID if you have a registered device. If you do not have a registered device, follow the steps shown on-screen to register a device.

5. Respond to the authentication request from PingID, and the PingFederate administrative console menu will display.

Result

> **Important**
>
> Access to the PingFederate administrative console can now only be performed by using directory user credentials, followed by a PingID authentication.

Enabling certificate-based authentication

You can enable certificate-based authentication in the PingFederate administrative console.

Before you begin

- Have a PingFederate username and password.
- Import the necessary client key and certificate into the web browser you use to access PingFederate.

About this task

To enable client-certificate authentication, PingFederate administrative users must import an X.509 key and a suitable certificate for user authentication into their web browsers. In addition, the corresponding root certificate authority (CA) certificates must be contained in the Java runtime or the PingFederate trusted store. Other setup steps, including designating user permissions, must be completed by using configuration files located in the `<pf_install>/pingfederate/bin` directory.

The roles configured in the properties file apply to both the administrative console and the administrative API.

Steps

1. Sign on to the PingFederate console as a user with permissions that include the **Crypto Admin** role.

1. Ensure the client-certificate's root CA and any intermediate CA certificates are contained in the trusted store, either for the Java runtime or PingFederate.

   > **Note**
   >
   > You can import a certificate to PingFederate in **Security** > **Certificate & Key Management** > **Trusted CAs**.

   > **Tip**
   >
   > You might want to click the Serial Number and copy the Issuer distinguished name (DN) to use in later steps.

2. In the `<pf_install>/pingfederate/bin/run.properties` file, change the value of the `pf.console.authentication` property as shown. `pf.console.authentication=cert`

3. In the `<pf_install>/pingfederate/bin/cert_auth.properties` file, enter the Issuer DN for the client certificate as a value for the property `rootca.issuer.x`, where `x` is a sequential number starting at `1`.

   > **Note**
   >
   > If you copied the Issuer DN after step 2, paste this value. For more information, see the comments in the file.

   The roles configured in the properties file apply to both the administrative console and the administrative API.

4. Repeat the previous step for any additional CAs as needed.

5. Enter the certificate user's Subject DN for the applicable PingFederate permission roles, as described in the properties file.

   > **Important**
   >
   > The configuration values are case-sensitive.

6. Repeat the previous step for all users as needed.

   > **Note**
   >
   > Other settings in the properties file are used to display the user's ID (Subject DN) in abbreviated form in the administrative console.

7. Start or restart PingFederate.

Configuring automatic connection validation

The intent of automatic multi-connection error checking is to verify that all configured connections have not been adversely affected by subsequent changes in supporting components.

About this task

Automatic multi-connection error checking occurs when you save certain supporting components, such as:

- SP adapters
- IdP adapters
- token processors
- token generators
- password credential validators
- identity store provisioners

As the number of connections and supporting components increases, so does the validation time. If you experience noticeable delays in saving adapters, token translators, password credential validators, or identity store provisioner, you can turn off automatic connection validation.

> **Note**
>
> When automatic connection validation is turned off, error checking is deferred until you access the connection lists on the administrative console. You can make configuration changes without being prompted to fix all dependency errors immediately, but remember that the configuration changes can potentially cause service disruption.
> For example, if you remove a configured source-attribute in an attribute fulfillment configuration in a connection, users will be unable to complete single sign-on (SSO) requests until you reconfigure such connections.
> When you access the **Connections** window, if the administrative console detects one or more dependency error conditions, it displays a visual cue to indicate the errors. To resolve each error, select the applicable connection and follow the on-screen instructions to modify the configuration.

This setting does not affect the validation of a connection being configured or modified. Also, individual connections are always validated automatically when you access them on the **Connections** window, regardless of the configuration of this setting.

To manage this setting for SP, go to **Applications > Integration > SP Connections** on the **SP Connections** window.

To manage this setting for IdP, go to **Authentication > Integration > IdP Connections** on the **IdP Connections** window.

Steps

- To turn off automatic multi-connection error checking, go to **System > Server > General Settings**, and select the **Disable Automatic Connection Validation** check box. This check box is not selected by default.

  For more information, see General settings.

  Result:

  After you select or clear the check box, the state of this setting is reflected on both the **SP Connections** window and the **IdP Connections** window.

Automating configuration migration

PingFederate provides a configuration-migration tool for scripting the transfer of administrative-console configurations and configuration property files from one PingFederate server to another.

The `configcopy` tool can migrate your configurations, such as from a test environment to production. It can also manage certificates for the target server.

> **Note**
>
> As of PingFederate 10.2, the `configcopy` tool has been deprecated and will be removed in a future release.

The command-line utility, `configcopy`, in the `<pf_install>/pingfederate/bin` directory, uses PingFederate's built-in Connection Management Service in conjunction with an internal Web Service to export and import connections and other configurations, and to obtain lists. For more information, see Connection Management Services.

> **Important**
>
> The Connection Management Service must be activated for both the source and target servers before you can use the `configcopy` tool. For more information, see Configuring service authentication.

> **Caution**
>
> For security reasons, you should disable the Connection Management Service whenever it is not in use.

Copying the key from the source to the target server

You must copy the key from the source server to the target server before you migrate data using the `configcopy` tool.

About this task

> **Note**
>
> As of PingFederate 10.2, the `configcopy` tool has been deprecated and will be removed in a future release.

To copy the key from the source to the target server, you copy the needed keys from the `pf.jwk` file on the source server and append it to the last key in the `pf.jwk` file on the target server, and then restart that target server. This step only needs to be completed before the first migration.

Steps

1. In your PingFederate installation on the source server, open the `pf.jwk` file in the `<pf_install>/pingfederate/server/default/data` directory.

2. Copy the key in the file.

   Make sure you copy the entire key JSON message, as in the following example.

   ```
   {"keys":[[.b]**\{"kty":"oct","kid":"j0PUEdAb95","k":"AGi8Lg_ewdl-_30Cx83kDMQE9oNlhgJSa_Pc4I8JTU8"}
   **]}
   ```

3. In your PingFederate installation on the target sever, open the `pf.jwk` file.

4. Insert a comma at the end of the last key in the file and append the source key.

   For example, if the `pf.jwk` on the target server reads as follows.

```
{"keys":[{"kty":"oct","kid":"wER9zEpaPe","k":"i0HQr9JmsqjAX4o_BQU1qGJzoLQI-nmwp8u3GyHzTB8"}]}
```

Insert the comma and the source key as shown.

```
{"keys":[{"kty":"oct","kid":"wER9zEpaPe","k":"i0HQr9JmsqjAX4o_BQU1qGJzoLQI-nmwp8u3GyHzTB8"},
 [.b]**\{"kty":"oct","kid":"j0PUEdAb95","k":"AGi8Lg_ewdl-_30Cx83kDMQE9oNlhgJSa_Pc4I8JTU8"}**]}
```

> **Note**
>
> This is a well-formed JSON document in one line.

5. Save the `pf.jwk` file and restart the target server.

6. If applicable, repeat the steps above for each target PingFederate server.

Administrative console migration

Use the `configcopy` tool to migrate data in the administrative console.

> **Note**
>
> As of PingFederate 10.2, the `configcopy` tool has been deprecated and will be removed in a future release.

For migrating data configured with the source server's administrative console, the `configcopy` tool performs these overall processing steps:

1. Retrieves specified connection and other configuration data (XML) from a source PingFederate server

2. Modifies the configuration with any changes required for the target environment, according to settings in one or more properties files, command-line arguments, or both

3. Imports the updated configuration into the PingFederate target server

The `configcopy` tool can perform these functions in real time, from server to server, or by using an intermediate file. The latter option is useful when both the source and target PingFederate servers are either not running at the same time or not accessible from the same operating system command window.

> **Important**
>
> For one-time configuration transfers from one version of PingFederate to a newer version, use a complete configuration archive, either with `configcopy` archive export/import commands, or manually through the administrative console, or the administrative API. Other `configcopy` commands are not supported for this purpose.

Operational capabilities include:

- Listing of source partner connections, adapter or STS token-translator instances, outbound-provisioning channels, or datastore connections.

  List commands include optional filter settings, when applicable.

- Copying one or more partner connections, outbound-provisioning channels, or instances of adapters or token translators.

- Copying one or more datastore connections.
- Copying server settings.
- Exporting and importing full configuration archives.

Copying configuration files

The `configcopy` tool supports copying configuration files containing runtime properties, including those needed for server clustering, that might have been manually customized for the source configuration and need to be migrated. The file-copy command can also copy the PingFederate internal, HSQLDB database when needed.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Managing Certificates

Administrators can use the `configcopy` tool to perform the following certificate-management tasks on the target PingFederate server:

- List source trusted certificate authorities (CAs) and target key aliases
- Copy one or all trusted CAs from the source server
- Create certificates
- Create Certificate Signing Requests (CSRs)
- Import CA-signed and PKCS-12 certificates

Using the migration tool

The migration tool, `configcopy`, can be used in conjunction with one or more property files to define the operational command and other parameters, including the source and target PingFederate servers, and to modify configuration settings as needed for the target environment.

About this task

> **Note**
>
> As of PingFederate 10.2, the `configcopy` tool has been deprecated and will be removed in a future release.

Property-file templates are available for each command option in the `<pf_install>/pingfederate/bin/configcopy_templates` directory.

> **Note**
>
> See the `README.txt` file in the `configcopy_templates` directory for a list of all commands and summary information. See the template files for parameters associated with each command or with use cases, as well as lists of Override Properties, which are configuration settings that can be modified in transit, where applicable.

Copies of the templates can be configured as needed and then used together, or combined into one file. Use the applicable file names as an argument when running `configcopy.bat` or `configcopy.sh`, depending on your operating system, for particular configurations, using the following command syntax.

(On Windows)

```
configcopy.bat -Dconfigcopy.conf.file=<properties_file1>; <properties_file2>;...
```

> **Note**
>
> When paths are included with the file names, you cannot use backslashes (`\`). Use forward slashes (`/`) or escape the backslash (`\\`).

(On Linux)

```
configcopy.sh -Dconfigcopy.conf.file=<properties_file1>:<properties_file2>:...
```

> **Note**
>
> The file separators are platform specific, corresponding to the syntax used for system-level path separators.

Also, you can specify any property values through command-execution arguments, using the following syntax

```
configcopy[.sh] -D<property>=<value> ...
```

where `<property>` is any property named in the properties file and `<value>` is the value. Command-line property designations take precedence over any values set in the properties file.

> **Note**
>
> Access to the Connection Management Service is password-protected. The usernames and passwords might be set in the properties file for both the source and target web services, and passwords can be obfuscated. If passwords are set in the properties file, they cannot be overridden using the command line. If a password is not set, the **`configcopy`** tool prompts for it. Usernames must always be supplied where applicable, either in the command line or in the properties file.

The **`configcopy`** utility generates its own log file, `configcopy.log`, located in the `<pf_install>/pingfederate/log` directory. You can control settings for this log, as needed, in the file `configcopy.log4j2.xml`, located in the `bin` directory.

> **Caution**
>
> Importing connections or other discrete configurations at the target server is not subject to the same rigorous data validation performed by the administrative console during manual configuration. Although some checks are made, it is possible to create invalid connections using the connection-migration process. Therefore, you should not use the `configcopy` tool to create settings at the target that do not exist at the source. For connections and other configurations copied separately, the tool is designed only for modifying the values of existing source settings to make them applicable to the target environment.
> To avoid errors and prevent unstable target configurations due to missing components or faulty cross-component references, such as invalid ID references from connection configurations to datastore configurations, adhere closely to the instructions provided in the following procedure.

Steps

1. Enable access to the Connection Management Service for both the source and target PingFederate servers. For more information, see Configuring service authentication.

2. Determine which component configurations need to be copied, including plugins.

   For example, connection configurations always reference either adapter or token-translator configurations, or both, and may reference datastore configurations. These are all separate configurations, and must be copied separately in conjunction with copying connection configurations, unless they already exist at the target.

   Server Settings, unless pre-configured at the target, also need to be copied over separately.

   Provisioning settings can be copied separately to update target connections, as needed.

1. Determine whether any configuration property files or other supporting files need to be copied.

2. Ensure necessary plugin JAR files are installed on the target server.

   The `configcopy` tool does not copy over these files, which include libraries for adapters, token translators, and JDBC or any custom database drivers.

   The JAR files are located in either `<pf_install>/pingfederate/server/default/deploy` or `<pf_install>/pingfederate/server/default/lib`.

3. On the target server, ensure that signing certificates, or certificates used for XML decryption, are already in place. For more information, see Certificate and key management.

   Private keys are not copied from server to server, public certificates can be copied. However, you can use `configcopy` to upload keys and certificates to the target server.

   Look for identifying information about the target keys so you can reference the certificates in connection-copy properties.

4. If you have not yet installed your organization's CA-issued SSL server certificate on both the target and source servers, you can do so with a `configcopy` command. Otherwise, use one of the following methods to ensure that `configcopy` can contact both servers:

   - (Recommended) Install the Issuer certificate for the PingFederate SSL certificate in a separately managed trust store. Then the location of the file can be specified when running `configcopy` using the property `configcopy.connection.trust.keystore`.

   - Install the Issuer certificate for the PingFederate SSL certificate into the trust store for the Java runtime where `configcopy` runs.

> **Note**
>
> If different SSL certificates are installed on the two servers, the **`configcopy`** tool must be able to trust both. In this case, both certificates must be installed in the trust store used by **`configcopy`**, or in the trust store for the Java runtime where **`configcopy`** runs.

5. Create properties files for the necessary commands, and for associated command-parameter values needed to copy the required configurations and any additional files.

   See the `README.txt` file and to the properties-file templates in the `<pf_install>/pingfederate/bin/configcopy_templates` directory.

   > **Note**
   >
   > This step and those following assume the use of properties files based on the templates provided. You can also use command-line parameters. For more information, see previous steps in this section.

6. If you are copying connections, override ID properties referencing adapter, datastores or other plugin configurations, as needed see [reuseStep2].

   > **Important**
   >
   > Ensure that the plugin configurations are either previously defined at the target or are part of the same **`configcopy`** process used to copy the connections that depend on them.

7. Create a script or run a command, or command series, that executes **`configcopy`** for each of the prepared properties files.

   See the previous discussion for syntax requirements, or the `README.txt` file.

Outbound provisioning CLI

PingFederate provides a command-line interface (CLI) to help manage automated outbound provisioning at identity provider (IdP) sites.

Administrators can use the CLI to view the status of user provisioning, either globally or one provisioning channel at a time, and to rectify unusual situations where provisioning at the service provider (SP) might be out of sync with the enterprise user store.

The CLI tool, `provmgr.bat` or `provmgr.sh`, is located in the directory `<pf_install>/pingfederate/bin`. The tool interacts with the PingFederate internal datastore to maintain provisioning synchronization between the LDAP user store and the target service.

The tool creates its own log file, located at `<pf_install>/pingfederate/log/provmgr.log`. You can control settings for this log, as needed, in the `<pf_install>/pingfederate/bin/provmgr.log4j2.xml` file.

The following table describes the available global and channel-specific command arguments.

Command argument	Description
Global options	

Command argument	Description
--help	Describes the available options. The help also displays if you run the command with no arguments.
--show-channels	Lists all channels in a table format, showing for each: • ID: A numeric channel ID (channel-specific commands need this ID) • Name: The channel name • Connection ID • Status: active | inactive (both the connection and the channel status are shown) • User count/dirty-user-record count, such as `5000/12`, which means 5000 users and 12 dirty records • Source, as LDAP URL • Target code
--show-nodes	Shows all the provisioning-server nodes with their status and the last timestamp. Applicable only when failover provisioning is configured in the `<pf_install>/pingfederate/bin/run.properties` file.
--force-node-backup Use with node number: -n <node ID>	Sets the provisioner mode to FAILOVER for the associated PingFederate server node.
Channel-specific options ⓘ **Note** With each command, specify the channel with the `-c <channel-id-number>` argument. For example: `provmgr -c 1 --show-source` You can determine channel ID numbers by using the global command `provmgr --show-channels`.	
--reset-group-timestamp	Deletes the user-group timestamp, which forces the provisioner to process the provisioning group on the next cycle, even if the timestamp on that group did not actually change. Depending on your LDAP server and administrative practices, you might want to schedule this command to run periodically to catch up with any users that may have been deleted, rather than deactivated, in the directory server. Some directory servers do not update the group timestamp for deleted users. **Important** You should rarely need this option if users are deactivated rather than deleted. If you do need it, you might want to schedule it when other network activity is low.

Command argument	Description
--reset-attribute-sync	Sets the attribute sync timestamp to 1, which forces the provisioner to look at all users for changes, not only those that have a newer timestamp on their LDAP entry. **Important** This is rarely needed and might consume considerable network resources, depending on the number of users. If it is needed, you might want to schedule it when other network activity is low.
--reset-values-hash	Removes the values hash for all users. The database stores a hash of attribute values for users to determine whether any values have been changed. This argument forces users that have a newer timestamp on their LDAP entry to be updated at the service provider, regardless of the actual field values. However, users whose recorded timestamp is unchanged are not updated.
--reset-all	Equivalent to using all three of the previous arguments.
--show-dirty-records	Lists all users or groups that have not been provisioned or updated at the SP site. This option is rarely needed and might consume considerable network resources, depending on the number of users. If it is needed, you might want to schedule it when other network activity is low.
--show-dirty-group-records	List groups that have not been provisioned or updated at the SP site.
--show-dirty-user-records	List all users that have not been provisioned or updated at the SP site.

--show-group
--show-user
Use with:
-u <provider name>
Or:
-g <LDAP GUID>
	Shows all internal database fields related to the specified user or group, including transitory mapping fields, which are fields waiting to be pushed to the SP. For a user, shows all LDAP attributes retrieved from the directory server.
	Note
	You can obtain user or group names and GUIDs for dirty records, as needed, using any of the `--show-dirty-*` options, described above.
	The LDAP GUID, if used and if it is binary, should be entered in hexadecimal format, as shown in log files.
	`provmgr.sh --show-user -u user@example.com`
	`provmgr.sh --show-user -g ffd448643f812b43a0bee2504173f0`

--clear-dirty-records	Clears the dirty flag on all records.
--clear-dirty-group-records	Clears the dirty flag on all group records.
--clear-dirty-user-records	Clears the dirty flag on all user records.
--delete-dirty-records	Removes all dirty records from the internal store.

Command argument	Description
--delete-dirty-group-records	Removes all dirty group records from the internal store.
--delete-dirty-user-records	Removes all dirty user records from the internal store.
--delete-all --delete-all-users	The `delete-all` parameter removes all users and groups from the internal store and deletes the provisioning group timestamp and the last attribute-sync timestamp. The `delete-all-users` parameter deletes users and timestamps but retains groups. The effect of either command is to reset the channel to its initial state for user provisioning. All user metadata is lost and provisioning for the channel will start from the beginning, picking up all users, and groups if deleted, and pushing them to the SP when the synchronization frequency interval has expired. The synchronization frequency interval is defined on **System > Server > Protocol Settings > Outbound Provisioning**. **Important** You should rarely need these options. If needed, you might want to schedule the operation when other network activity is low.
--show-target	Displays the target configuration.
--show-source	Displays all source LDAP configuration parameters, including settings and location.

Customizable user-facing pages

PingFederate supplies HTML templates, located in the `<pf_install>/pingfederate/server/default/conf/template` directory, to provide information to the end-users or to request user input when processing their requests.

The PingFederate HTML templates utilize the Velocity template engine, an open-source Apache project. For more information about Velocity, please refer to the Velocity project documentation on the Apache website at https://velocity.apache.org.

You can modify most of these pages in a text editor to suit the particular branding and informational needs of your PingFederate installation. CSS and images for these pages are included in the `template/assets` subdirectory. Each page contains both Velocity constructs and standard HTML. The Velocity engine interprets the commands embedded in the template page before the HTML is rendered in the user's browser. At runtime, PingFederate supplies values for the Velocity variables used in the template.

> **Note**
>
> You can develop and deploy your own tools using the Velocity tools framework. For a full list of available tools, see Tools Usage Summary in the Apache Velocity documentation.

Each template contains specific variables that can be used for rendering the associated web page. You can see the variables and usage examples in the comments of each template.

The following table describes variables that are available across all templates.

Variable	Description and Usage
utils- utility class	The utility method to display JSON String arrays. `$utils.toJsonArray(Collection<Object>)` - Use this method to convert a collection into a JSON string.
$escape	A utility class that can be used to escape String variables inserted into the template, such as `$escape.escape($client.name)` where *$client.name* is one of the variables available in the `oauth.approval.page.template.html` template file. Use `$escape.forJavaScript($variable)` when passing String variables into a JavaScript code block or an event handler within a template, such as `window.location.replace("$escape.forJavaScript($wreply)")` as seen in the `sourceid-wsfed-idp-signout-cleanup-template.html` template file. **Important** Use the *$escape* variable to escape external data, such as request parameters, to mitigate the risk of potential cross-site scripting (XSS) attacks.
$HttpServletRequest	A Java object instance of `javax.servlet.http.HttpServletRequest`. Used to add additional knowledge about the request that is otherwise unavailable in the template, such as the *User-Agent* HTTP header.
$HttpServletResponse	A Java object instance of `javax.servlet.http.HttpServletResponse`. Used to modify the response in the template, such as setting additional browser cookies.
$locale	A Java object instance of `java.util.Locale` that represents a user's country and language. Used to customize the end-user experience. For example, the locale is used to display content in the user's preferred language.
$CurrentPingFedBaseURL	The host name found in the request, provided that it matches either the PingFederate's base URL or one of the configured virtual host names.
$PingFedBaseURL	The PingFederate base URL. For most deployments, use the *$CurrentPingFedBaseURL*variable instead of the *$PingFedBaseURL* variable.
$templateMessages	Used to localize messages in the template, based on user's Locale, an instance of `com.pingidentity.sdk.locale.LanguagePackMessages`. For more information, see the Javadoc for the `LanguagePackMessages` class in the directory `<pf_install>/pingfederate/sdk/doc`.
$TrackingId	The user's session tracking ID.

The following describes variables that are available on some templates.

Variable	Description
$entityId	The entity ID (connection ID) of the SP connection used in this SSO transaction.

Variable	Description
$connectionName	The name of the SP Connection used in this SSO transaction.
$client_id	The ID of the OAuth client used in this transaction.
$spAdapterId	The SP Adapter ID used in this transaction.
$baseUrl	The base URL of PingFederate instance.
$adapterId	The IdP Adapter ID used in this transaction.
$oidcUiLocales	The value of the OpenID Connect `ui_locales` parameter that conveys the user's preferred languages and scripts for the user interface.
$extendedProperties	The extended properties defined on either the connection or OAuth client.
$userAttributes	The user's display name, email address, and other user-specific data retrieved from the template type used in this transaction. The $userAttributes variable represents the attributes associated with a user's identity and enables the retrieval of user-specific information across templates. In Local Identity Profile (LIP)-related templates, the attribute names of *$userAttributes* are derived from the data store mapping configured in **Authentication > Policies > Local Identity Profiles > Data Store Configuration > Data Store Mapping**. For example, *userAttributes.email* is an attribute in this context. In Adapter-related templates, the attribute names of *$userAttributes* are based on the configured contract in the LDAP-type password credential validator (PCV). For non-LDAP PCVs, the attribute names are derived from the implementation of the SDK method, `ResettablePasswordCredential.findUser()` . The following attributes are commonly used: • *userAttributes.userName* • *userAttributes.givenName* • *userAttributes.mail* • *userAttributes.phone* • *userAttributes.pingid* • *userAttributes.mailVerified*
$grantAttributes	The attributes of the grant used in this transaction.

Important

Changing Velocity or JavaScript code is not recommended.

At runtime, the user's browser is directed to the appropriate page, depending on the operation being performed and where the related condition occurs. For example, if a single sign-on (SSO) error occurs during identity provider (IdP)-initiated SSO, the user's browser is directed to the IdP's SSO error-handling page.

Applications can override the PingFederate server-hosted pages provided specifically for SSO and single logout (SLO) errors by specifying a URL value in the relevant application endpoint's `InErrorResource` parameter. Administrators can override SSO and SLO success pages by specifying default URLs on the **SP Default URLs** window (**Applications > Integration > SP Default URLs**) or the **IdP Default URL** window (**Authentication > Integration > IdP Default URL**).

The Velocity templates retrieve titles and other text from a message-property file, `pingfederate-messages.properties`, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory. You can also localize these messages using the PingFederate localization framework.

Note

If you have a clustered PingFederate environment, copy the customized, and localized, templates to each node.

Strict content security policy for HTML templates

Content security policy (CSP) is a security feature that protects against cross-site scripting (XSS) attacks by controlling what resources a web page can load. When creating customizable user-facing pages, ensure that your HTML templates follow a strict CSP to prevent the execution of any unsafe or unauthorized scripts and resources on the user's browser. For more information, see Content Security Policy (CSP) in the Mozilla developer documentation.

PingFederate enforces a CSP on its HTML templates through the `<meta>` HTML element. The CSP configured though this element varies depending on the template and the use-cases it supports. You can modify the CSP to work with your template customizations and security requirements. The `$CSPNonce` Velocity variable is available on all templates, and you can use this variable to allow inline scripts and styles by including the nonce in `<script>` and `<script>` HTML tags.

Related links

- HttpServletRequest (javaee.github.io/javaee-spec/javadocs/javax/servlet/http/HttpServletRequest.html)
- HttpServletResponse (javaee.github.io/javaee-spec/javadocs/javax/servlet/http/HttpServletResponse.html)

IdP user-facing pages

PingFederate has a variety of customizable user-facing page templates that apply to identity provider (IdP) pages. The templates are organized by HTML Form Adapter, Kerberos Adapter, single sign-on (SSO), single logout (SLO), WS-Federation, and OpenID Connect (OIDC).

HTML Form Adapter

Page title and template file name	Purpose	Type	Action
Sign On or Choose an Account identifier.first.template.html	Prompts a user to provide their username when an Identifier First Adapter instance is invoked to handle a sign-on request.	Normal	User input required
Sign On html.form.login.template.html	Displays a customizable user sign-on form when an HTML Form Adapter instance is invoked to handle a sign-on request. If the invoked HTML Form Adapter instance is associated with a local identity profile configured to support authentication via third-party identity providers, the sign-on page will display those identity providers. This is a core HTML template.	Normal	User input required
Change Password html.form.change.password.template.html	Displayed when a user attempts to change their password through the HTML Form Adapter.	Normal	User input required
Change Password html.form.message.template.html	Displayed when a user successfully changes their password. This is a core HTML template.	Normal	User input required
Password Expiring html.form.password.expiring.notification.template.html	Displayed to warn an authenticated user that the password associated with the account is about to expire. This is a core HTML template.	Normal	User input required
Password Management System Message html.form.message.template.html	Displayed when a user is redirected to a password management system to change their password. This is a core HTML template.	Normal	User input required

Page title and template file name	Purpose	Type	Action
Account Recovery forgot-password.html	Displayed when a user attempts to reset their password through the HTML Form Adapter. If the user enters a username in the sign-on form, the username carries over to this form. Otherwise, the user must enter their username to begin the self-service password reset process.	Normal	User input required
Account Recovery forgot-password-resume.html	Displayed to prompt a user to enter the one-time password sent through a notification or to notify a user to refer to the notification for password reset instructions. This template is applicable when the password reset type is **Email One-Time Link**, **Email One-Time Password**, or **Text Message** for the invoked HTML Form Adapter instance.	Normal	User input required
Reset Your Password forgot-password-change.html	Displayed to prompt a user to define a new password.	Normal	User input required
Account Recovery forgot-password-success.html	Displayed when a user successfully resets their password.	Normal	User input required
Account Recovery forgot-password-error.html	Displayed when a password reset attempt fails.	Error	None
Unlock Your Account account-unlock.html	Displayed when a user successfully unlocks their account through the HTML Form Adapter. This page also prompts the user to retain the current password, or reset it.	Normal	User input required

Page title and template file name	Purpose	Type	Action
Security Question html.form.login.challenge.template.html	Displays a configurable challenge form for two-step authentication. For example, this template can be used to create a RADIUS challenge form when using the RADIUS Username/Password Credential Validator. This is a core HTML template.	Normal	User input required
User Consent consent-form-template.html	Displayed when a request requires a user's consent for an SSO to an SP.	Normal	User input required
Logout Confirmation idp.slo.confirm.page.template.html	Displayed when a user initiates a logout request. Applicable only if such confirmation is required, as configured on the **Authentication > Integration > IdP Default URL** window.	Normal	User input required
Sign Off idp.logout.success.page.template.html	Displayed when a user successfully signs off in a configuration where the **Logout Path** field is configured but the **Logout Redirect** field is not.	Normal	None
Create Your Account local.identity.registration.html	Displays a configurable challenge form for two-step authentication.Displayed when a user requests to register for a local account. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that is configured to support self-service registration.	Normal	User input required

Page title and template file name	Purpose	Type	Action
Manage Your Profile local.identity.profile.html	Displayed when an authenticated user accesses the profile management endpoint. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that is configured to support self-service profile management.	Normal	User input required
Email Verification local.identity.email.verification.sent.html	Displays a notification that an email ownership verification message has been sent when an authenticated user accesses the email ownership verification endpoint. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that is configured to offer users the opportunity to verify the ownership of the email address associated with the accounts.	Normal	None
Email Verified local.identity.email.verification.success.html	Displays a confirmation that the user has successfully verified the ownership of the email address associated with the account. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that is configured to offer users the opportunity to verify the ownership of the email address associated with the accounts.	Normal	None

Page title and template file name	Purpose	Type	Action
Email Verification Error local.identity.email.verification.error.html	Displays that the user failed to verify the ownership of the email address associated with the account. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that is configured to offer users the opportunity to verify the ownership of the email address associated with the accounts.	Error	User can request another verification email by accessing the email ownership verification endpoint or the profile management page (if enabled). Authentication is required. Alternatively, the user can contact their IT administrators for further assistance.
Username Recovery username.recovery.template.html	Displays to prompt the user to enter an email address to recover the username associated with the account. Applicable only if the invoked HTML Form Adapter instance is configured to support self-service username recovery.	Normal	User input required
Username Recovery username.recovery.info.template.html	Displays to notify the user to retrieve the notification message with the recovered username. Applicable only if the invoked HTML Form Adapter instance is configured to support self-service username recovery.	Normal	User should retrieve the notification message with the recovered username.

Kerberos Adapter

Page title and template file name	Purpose	Type	Action
Error kerberos.error.template.html	Displays an error page to provide standardized information to the end user when the authentication attempt fails.	Error	Consult log
(No title) meta.refresh.template.html	Facilitates the failover mechanism from a Kerberos Adapter instance to the next phase when it is part of a Composite Adapter instance configuration or an authentication policy.	Normal	None

Single sign-on and logout

Page title and template file name	Purpose	Type	Action
Select Authentication System sourceid-choose-idp-adapter-form-template.html	Displayed when multiple authentication sources are applicable and no preference is submitted as part of the request.	Normal	User input required
Sign On Error idp.sso.error.page.template.html	Displayed when IdP-initiated or adapter-to-adapter SSO fails and no other SSO error landing page is specified.	Error	Consult log and web developer
Sign Off Successful idp.slo.success.page.template.html	Displayed when an SLO request succeeds and no other SLO success landing page is specified.	Normal	None
Sign Off Error idp.slo.error.page.template.html	Displayed when an SLO request fails and no other SLO error landing page is specified.	Error	User should close the browser

WS-Federation and OpenID Connect

Page title and template file name	Purpose	Type	Action
Working . . . sourceid-wsfed-http-post-template.html	Used to auto-submit a WS-Federation assertion to the SP. If JavaScript is disabled, the user is prompted to click a button to POST the assertion directly. This page is normally not displayed if JavaScript executes properly.	Normal	None
Signing off. . . sourceid-wsfed-idp-signout-cleanup-invisible-template.html	WS-Federation and OIDC client IdP sign-out processing page. No HTML is rendered in the browser.	Normal	None
Sign Off Successful sourceid-wsfed-idp-signout-cleanup-template.html	Indicates user signed out of the IdP under the WS-Federation protocol and lists each successful SP logout, when applicable. Also displays when an OIDC client sends a logout request to the `/idp/startSLO.ping` endpoint to initiate an Asynchronous Front-Channel Logout process.	Normal	None

SP user-facing pages

PingFederate has a variety of customizable user-facing page templates that apply to service provider (SP) pages. Each template contains specific variables that can be used for rendering the associated web page.

Account linking

Page title and template file name	Purpose	Type	Action
Link Your Account LocalIdPasswordLookup.form.template.html	Used to authenticate a user at the SP when an account link needs to be established.	Normal	None
Account Unlinked TerminateAccountLinks.page.template.html	Communicates a user's successful defederation operation.	Normal	None

Single sign-on and logout

Page title and template file name	Purpose	Type	Action
Select Identity Provider sourceid-saml2-idp-selection-template.html	The user requested SP-initiated single sign-on (SSO), but the identity provider (IdP) partner was not specified in the appropriate query parameter or cookie. This page allows the user to select the IdP manually. Based on the user's selection, the server redirects the browser to the appropriate IdP partner's SSO service.	Normal	User must make selection
Please Specify Target sp.sso.success.page.template.html	Displayed when an SSO request succeeds but no target-resource parameter is specified by the incoming URL, and no default URL is set on the **Applications > Integration > SP Default URLs** window.	Error	Consult web developer or specify default URL
Sign On Error sp.sso.error.page.template.html	Displayed when SP-initiated SSO fails, or IdP-initiated SSO fails on the SP side, and no other SSO error landing page is specified.	Error	Consult log and web developer
Sign Off Successful sp.slo.success.page.template.html	Displayed when a single logout (SLO) request succeeds and no other SLO success landing page is specified.	Normal	None
Sign Off Error sp.slo.error.page.template.html	Displayed when an SLO request fails and no other SLO error landing page is specified.	Error	User should close the browser

WS-Federation

Page title and template file name	Purpose	Type	Action
Signed Off sourceid-wsfed-sp-signout-cleanup-template.html	Displays the user's sign-out status.	Normal	None
Unable to Authenticate sourceid-wsfed-idp-exception-template.html	Displayed when an authentication challenge fails during WS-Federation processing.	Error	Consult log and web developer

Either IdP or SP user-facing pages

PingFederate allows customizable user-facing windows that can be applied to both the identity provider (IdP) and service provider (SP) pages.

Page title and template file name	Purpose	Type	Action
Sign On AbstractPasswordIdpAuthnAdapter.form.template.html	Challenges user for credentials when authentication can take place via HTTP Basic authentication or an HTML form, depending on the operational mode.	Normal	User must sign on
Submit Form form.autopost.template.html	Whenever the server posts a form, this template is used to auto-submit the form. If JavaScript is disabled, the user is prompted to click a button to post the form manually. This page is normally not displayed if JavaScript executes properly.	Normal	None
Multiple Sign-On Delay speed.bump.template.html	Displayed to indicate that simultaneous single sign-on requests from multiple browser tabs are in progress.	Normal	User can switch to the browser tab that is actively waiting for the user to complete the authentication requirement or resubmit the request.
Error general.error.page.template.html	Indicates that an unknown error has occurred and provides a error reference number and, optionally, an error message.	Error	Consult log Contact Ping Identity Support if unresolved

Page title and template file name	Purpose	Type	Action
Error generic.error.msg.page.template.html	General error, with error code.	Error	Consult log and check configuration Contact Ping Identity Support if unresolved
Error http.error.page.template.html	Indicates that an HTTP error has occurred and provides the HTTP status code.	Error	Consult log Contact Ping Identity Support if unresolved
Page Expired state.not.found.error.page.template.html	Displayed when simultaneous single sign-on (SSO) requests from multiple tabs using the same PingFederate cookie cause a user session to be overwritten or deleted and remaining requests attempt to retrieve the state fail.	Error	None

OAuth user-facing pages

The PingFederate OAuth authorization server provides five windows that are presented to end-users, or resource owners, during certain OAuth transactions. You can customize and brand these windows as needed..

Page title and template file name	Purpose	Message type	Action
Client Access oauth.access.grants.page.template.html	Provides a means for the end users, or resource owners, to revoke persistent access grants.	Normal	User input required

Page title and template file name	Purpose	Message type	Action
Request for Approval oauth.approval.page.template.html	Advises resource owners that their information is being requested by the identified OAuth client when the default, internal, consent user interface is used. Resource owners can approve or deny individual scopes. Consent approval is applicable to the Device Authorization, Implicit, and Authorization Code grant types. For the latter two, PingFederate might prompt once at first or repeatedly depending on the **Reuse Existing Persistent Access Grants for Grant Types** setting in **System > OAuth Settings > Authorization Server Settings**.. In addition, the OAuth client configuration provides an option to bypass this approval page entirely, as needed for trusted clients. When applicable, select the **Bypass Authorization Approval** check box in the client configuration window. When an external consent user interface is used, PingFederate does not make use of this template file.	Normal	User input required
Connect a device (user code prompt) oauth.device.user-code.page.template.html	This page appears for the OAuth device authorization grant type. It allows resource owners to identify an authorization session that was initiated by the device client. This page appears after the resource owner goes to the OAuth verification URL and logs in. The user types the user code that they received from the device client, and then clicks **Submit**.	Normal	User input and confirmation required

Page title and template file name	Purpose	Message type	Action
Connect a device (pre-populated user code prompt) oauth.device.user-code-confirm.page.template.html	This page appears for the OAuth device authorization grant type. It allows resource owners to identify an authorization session that was initiated by the device client. This page appears after the resource owner goes to the OAuth verification URL and logs in. The user confirms the pre-populated user code by clicking **Confirm**.	Normal	User confirmation required
Connect a device (result) oauth.device.messages.page.template.html	This page appears for the OAuth device authorization grant type. It advises resource owners whether the OAuth device authorization was successful and provides any relevant error messages. By default, this page does not link to any other pages.	Normal	No action

Customizable email notifications

PingFederate delivers messages to administrators and end-users based on customizable notification publisher settings.

Each component that is capable of triggering or handling events can use a different notification publisher instance to deliver its messages. For example, you can select an SMTP Notification Publisher instance to deliver messages to your end users in an HTML Form Adapter instance and another SMTP Notification Publisher instance to deliver licensing messages to your fellow administrators.

When a component is configured to deliver its messages based on an SMTP Notification Publisher instance configuration, PingFederate creates notification messages based on template files located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory. Each template file is a combination of variables and HTML codes. You can modify these template files in a text editor to suit the particular branding requirements, as needed.

Each template contains specific variables that can be used for rendering the associated web page. You can see the variables and usage examples in the comments of each template.

Variables available across all templates

Variable	Description and Usage
utils- utility class	The utility method to display JSON String arrays. `$utils.toJsonArray(Collection<Object>)` - Use this method to convert a collection into a JSON string.

Variable	Description and Usage
$escape	A utility class that can be used to escape String variables inserted into the template, such as `$escape.escape($client.name)` where *$client.name* is one of the variables available in the `oauth.approval.page.template.html` template file. Use `$escape.forJavaScript($variable)` when passing String variables into a JavaScript code block or an event handler within a template, such as `window.location.replace("$escape.forJavaScript($wreply)")` as seen in the `sourceid-wsfed-idp-signout-cleanup-template.html` template file. **Important** Use the *$escape* variable to escape external data, such as request parameters, to mitigate the risk of potential cross-site scripting (XSS) attacks.
$HttpServletRequest	A Java object instance of `javax.servlet.http.HttpServletRequest`. Used to add additional knowledge about the request that is otherwise unavailable in the template, such as the *User-Agent* HTTP header.
$HttpServletResponse	A Java object instance of `javax.servlet.http.HttpServletResponse`. Used to modify the response in the template, such as setting additional browser cookies.
$locale	A Java object instance of `java.util.Locale` that represents a user's country and language. Used to customize the end-user experience. For example, the locale is used to display content in the user's preferred language.
$CurrentPingFedBaseURL	The host name found in the request, provided that it matches either the PingFederate's base URL or one of the configured virtual host names.
$PingFedBaseURL	The PingFederate base URL. For most deployments, use the *$CurrentPingFedBaseURL*variable instead of the *$PingFedBaseURL* variable.
$templateMessages	Used to localize messages in the template, based on user's Locale, an instance of `com.pingidentity.sdk.locale.LanguagePackMessages`. For more information, see the Javadoc for the `LanguagePackMessages` class in the directory `<pf_install>/pingfederate/sdk/doc`.
$TrackingId	The user's session tracking ID.

Variables available on some templates

Variable	Description
$entityId	The entity ID (connection ID) of the SP connection used in this SSO transaction.
$connectionName	The name of the SP Connection used in this SSO transaction.

Variable	Description
$client_id	The ID of the OAuth client used in this transaction.
$spAdapterId	The SP Adapter ID used in this transaction.
$baseUrl	The base URL of PingFederate instance.
$adapterId	The IdP Adapter ID used in this transaction.
$oidcUiLocales	The value of the OpenID Connect `ui_locales` parameter that conveys the user's preferred languages and scripts for the user interface.
$extendedProperties	The extended properties defined on either the connection or OAuth client.
$userAttributes	The user-specific data, such as the user's display name or email address, retrieved from the password credential validator (PCV) used in this transaction.
$grantAttributes	The attributes of the grant used in this transaction.

> **Note**
>
> If you have a clustered PingFederate environment, copy the customized templates to each node.

> **Tip**
>
> You can also configure a component to use an Amazon SNS Notification Publisher instance to deliver notification messages. If so, refer to Configuring an Amazon SNS Notification Publisher instance and Event types and variables for more information about this messaging model and the handling of notification messages.

Related links

- Configuring an SMTP Notification Publisher instance

Local administrative account management events

PingFederate generates a variety of email notifications for local administrative account management events.

Notification for account management events is configurable from **System > Server** in the **Administrative Accounts**window.

> **Note**
>
> Account management events are only applicable when native authentication is enabled for the administrative console, the administrative API, or both in the `<pf_install>/pingfederate/bin/run.properties` file. If you are using an alternative console authentication, notifications, if any, such as password changes, are handled by the third-party system.

Email subject and template file name	Event	Action
PingFederate Notification Settings Change Notification message-template-notifications.html	An administrator has turned off the **Notify Administrator of Account Changes** option. PingFederate generates a notification message to all administrators. The message includes the username of the administrator who made the change.	Ensure this change is approved and legitimate.
PingFederate Email Change Notification message-template-email.html	An administrator's email address has been updated by another administrator. PingFederate generates a notification message to the previous email address and another notification to the new email address. The message includes the username of the administrator who made the change.	Ensure this change is approved and legitimate.
PingFederate Password Change Notification message-template-password.html	An administrator's password has been changed. PingFederate generates a notification to the administrator whose password has been changed. The message includes the username of the administrator who made the change.	Ensure this change is approved and legitimate.

Related links

- Enabling notification messages for account management events

Certificate events

PingFederate sends email notifications for the creation, update, and expiration of certificates.

PingFederate also sends email notifications for the creation and activation of new pending certificates when automatic rotation for self-signed certificates is enabled in the **Security > Certificate & Key Management > Signing & Decryption Keys & Certificates** window. It sends notifications to the email address that has been configured for certificate events on the **Runtime Notifications** window.

Email subject and template file name	Event	Action
A PingFederate Certificate Is About to Expire message-template-cert-warning.html	A certificate is about to expire. PingFederate generates a notification based on settings defined on the **Runtime Notifications** window. The message includes the details of the certificate and the connections associated with it.	Create a new certificate and work with the applicable partners to update the expiring certificate. If a self-signed certificate is used for signing or decryption, consider enabling certificate rotation while creating the new certificate. For a clustered PingFederate environment, replicate the configuration using the administrative console.
A PingFederate Certificate Has Expired message-template-cert-expire.html	A certificate expired. PingFederate generates a notification upon the expiration of a certificate. The message includes the details and the connections associated with the certificate.	Create a new certificate and work with the applicable partners to update the expiring certificate. If a self-signed certificate is used for signing or decryption, consider enabling certificate rotation while creating the new certificate. For a clustered PingFederate environment, replicate the configuration using the administrative console.
A New PingFederate Certificate Has Been Created message-template-cert-rotation.html	A new pending certificate has been created for signing or decryption. PingFederate generates a notification when a new pending certificate is created. The message includes the details of the current certificate, the details of the new certificate, the activation date, and the connections that will be affected when the new certificate is activated.	Work with the applicable partners to update the expiring certificate. PingFederate supports providing metadata for Browser SSO connections. For a clustered PingFederate environment, replicate the configuration using the administrative console.
A PingFederate Certificate Has Been Updated message-template-cert-deactivation.html	A new certificate for signing or decryption is activated. PingFederate generates a notification when the new certificate is activated. The message includes the details of the new certificate and the affected connections.	None, unless the applicable partners have not been notified or configuration has not been replicated in a clustered PingFederate environment.

SAML metadata update events

When notification for SAML metadata update events is enabled in the **Runtime Notifications** window, PingFederate sends the notifications to the email address that has been configured for the event.

PingFederate supports automatic reloading of SAML metadata, which streamlines the maintenance of SAML connections.

Email subject and template file name	Event	Action
Your ${CONNECTION_NAME} ${SP_IDP} Connection in PingFederate Has Been Updated message-template-metadata-updated.html	PingFederate has updated a connection based on the partner's SAML metadata.	For a clustered PingFederate environment, replicate the configuration using the administrative console.
Please Review Your Updated ${CONNECTION_NAME} ${SP_IDP} Connection in PingFederate message-template-metadata-updated-out-of-sync.html	PingFederate has updated a connection based on the partner's SAML metadata. However, some settings are out of sync.	Review the settings that are out of sync and make changes as needed. For a clustered PingFederate environment, replicate the configuration using the administrative console.
Your ${CONNECTION_NAME} ${SP_IDP} Connection in PingFederate Is Out of Sync message-template-metadata-out-of-sync.html	A connection is out of sync with the changes found in the partner's SAML metadata.	Review the settings that are out of sync and make changes as needed. For a clustered PingFederate environment, replicate the configuration using the administrative console.
Your Metadata Couldn't Be Downloaded from ${METADATA_URL_NAME} message-template-metadata-url-notification.html	PingFederate failed to download the SAML metadata from the partner or could not validate the digital signature of the metadata.	Consult log. Verify and update the metadata URL and its corresponding verification certificate.
Your Metadata for PingFederate ${SP_IDP} Connection ${CONNECTION_NAME} Wasn't Found message-template-metadata-url-entity-id-missing.html	The partner's metadata URL did not return the expected metadata for a given connection.	Consult log. Verify and update the metadata URL.

Licensing events

There is a list of different licensing events and different actions you can take to resolve each event.

When you enable notifications for licensing events, PingFederate sends license expiry information to the recipient configured for the event on the **Runtime Notifications** window.

Tip

To check the details of your license, sign on to the administrative console, navigate to the user icon in the upper-right corner of the administrative console, and click **About** the list. The license summary displays in a pop-up browser window.

Important

If the license specifies an expiration date, the license expires at the beginning of that day.

Email subject and template file name	Event	Action
Your PingFederate License Is About to Expire message-template-warn.html	The current license is about to expire. The email notification is sent 60 days ahead of the expiration. Applicable to licenses without connection groups.	Contact sales@pingidentity.com to renew the license before the expiration.
Your PingFederate License Is About to Expire message-template-group-warn.html	One of the connection groups in the current license is about to expire. The email notification is sent 60 days ahead of the expiration. Applicable to licenses with connection groups.	Contact sales@pingidentity.com to renew the license before the expiration.
PingFederate License Expiration Notification message-template-grace.html	The current license expired. The email notification is sent upon the expiration. Applicable to licenses without connection groups.	Contact sales@pingidentity.com to renew the license.
Your PingFederate License Has Expired message-template-group-grace.html	One of the connection groups in the current license expired. The email notification is sent upon the expiration. Applicable to licenses with connection groups.	Contact sales@pingidentity.com to renew the license.
PingFederate Has Stopped Processing Requests message-template-shutdown.html	The current license and the grace period, if any, had lapsed. Applicable to licenses without connection groups.	Contact sales@pingidentity.com to renew the license.

HTML Form Adapter events

The HTML Form Adapter offers self-service account management tools for password management, account recovery, username recovery, and email ownership verification.

When you configure an HTML Form Adapter instance to deliver its messages based on an SMTP Notification Publisher instance configuration, administrators can optionally customize and localize the following template files with branding controls to provide a consistent brand experience to end users across multiple user populations.

Email subject and template file name	Event	Action
Password Change Notification message-template-end-user-password-change.html	A user's password has been changed successfully through an instance of the HTML Form Adapter.	Users should contact their IT administrators if they have not made any attempts to change their password.
Password Reset message-template-forgot-password-link.html	A user has initiated a self-service password reset request. Applicable when the password reset type for the invoked HTML Form Adapter instance is set to **Email One-Time Link**.	Users should contact their IT administrators if they have not made any attempts to reset their password.
Password Reset message-template-forgot-password-code.html	A user has initiated a self-service password reset request. Applicable when the password reset type for the invoked HTML Form Adapter instance is set to **Email One-Time Password** or **Text Message**.	Users should contact their IT administrators if they have not made any attempts to reset their password.
Password Reset Completed message-template-forgot-password-complete.html	A user's password has been reset successfully through an instance of the HTML Form Adapter. Applicable when the password reset type for the invoked HTML Form Adapter instance is set to any method except **None**.	Users should contact their IT administrators if they have not made any attempts to reset their password.
Account Unlocked message-template-account-unlock-complete.html	A user account has been unlocked successfully through an instance of the HTML Form Adapter. Applicable when the **Account Unlock** option is enabled for the invoked HTML Form Adapter instance.	Users should contact their IT administrators if they have not made any attempts to unlock their account.
Account Unlock Email Template		
Password Reset Failed message-template-forgot-password-failed.html	An attempt to reset the end-user's password has failed.	Users should contact their IT administrators if they have not made any attempts to reset their password, or to seek help in resetting their password.

Email subject and template file name	Event	Action
Email Verification message-template-email-ownership-verification.html	A user has provided an email address on the registration page, updated an existing email address with a new one on the profile management page, requested a resend of the verification email on the profile management page, or accessed the email ownership verification endpoint. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that has been configured to offer users the opportunity to verify the ownership of the email address associated with the accounts.	Users should contact their IT administrators if they have not made any attempts to update the email address associated with their accounts.
Username Recovery message-template-username-recovery.html	A user has initiated a self-service username recovery request and provided an email address through an instance of the HTML Form Adapter. If PingFederate can locate the user record using such email address and other requirements are met, PingFederate uses this template to generate an email message containing the recovered username and sends it to the user at the email address provided by the user. Applicable only if the invoked HTML Form Adapter instance is configured to support self-service username recovery.	Users should contact their IT administrators if they have not made any attempts to recover the username associated with their accounts.

Related links

- Configuring an HTML Form Adapter instance
- HTML Form Adapter advanced fields
- Localizing messages for end users

Customizable text message

You can customize text messages in PingFederate for a unique experience.

If you have configured your self-service password reset and, optionally, your self-service account unlock to use the text message option, you can customize and localize the text message for a unique experience. The default message is stored as a property in the `<pf_install>/pingfederate/server/default/conf/language-packs/pingfederate-sms-messages.properties` file.

Note

If you have a clustered PingFederate environment, copy the customized, and localized, templates to each node.

Localizing messages for end users

PingFederate supports localization for several types of messages for end users.

About this task

Administrators can localize the following message types:

- on-screen messages
- email messages
- text messages (SMS)
- authentication API error messages

The English contents for each message type are in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

Message type	Default message file
On-screen messages	`pingfederate-messages.properties`
Email messages	`pingfederate-email-messages.properties`
Text messages (SMS)	`pingfederate-sms-messages.properties`
Authentication API error messages	`authn-api-messages.properties`

Steps

1. Create a copy of the associated default message file in the `language-packs` directory.
2. Provide translated text in place of English and then append the standard language tag to the base file name, as indicated in the browser settings.

   For example, to localize the user-facing window messages in French, rename the translated copy of the localization file to `pingfederate-messages_fr.properties`.

   If the system language of the PingFederate server is not English, copy the default (English) message file and append `_en` at the end of the file name for any English users, such as from `pingfederate-messages.properties` to `pingfederate-messages_en.properties`, translate the default message file for the local users, and provide additional translations as needed.
3. If you want to include a region, append the capitalized abbreviation to the standard language tag with an underscore between them.

   For example, to localize the text messages in Canadian French, rename the translated copy to `pingfederate-sms-messages_fr_CA.properties`.

> **Note**
>
> The capitalization and underscore usage might not correspond to the way regions are listed in browser settings. However, the usage is required by the Java-based localization implementation.

4. If you have a clustered PingFederate environment, copy the localized message files to each node.

Result

For on-screen and email messages, developers can also customize the look and feel of the templates by using localization variables in logic statements to control fonts, color, and other style elements. For more information, see the template files for examples.

> **Tip**
>
> To maximize performance, PingFederate caches localized UI strings on start-up. For testing new localization implementations, an administrator can temporarily turn off caching by changing the value of the `cache-language-pack-messages` element to `false` in the `<pf_install>/pingfederate/server/default/data/config-store/locale-options.xml` file. Return the value to `true` when testing is complete. You must restart the server after changes to configuration files.

Locale overrides by cookies

An administrator or web developer might want to provide end-users a means of overriding browser language preferences temporarily by setting cookies, such as creating a company web portal link for users to click instead of manually changing their browser options.

By default, the PingFederate localization framework supports overriding the locale using a cookie named `pf-accept-language`. The cookie value must conform to the guidelines defined under IETF BCP 47. For more information about the Java core method that PingFederate uses to parse the cookie, see .oracle.com/en/java/javase/11/docs/api/java.base/java/util/Locale.html//[Locale.forLanguageTag(String languageTag)^].

This locale-override behavior is the default implementation of the Java interface, `LocaleOverrideService`, defined in the PingFederate SDK. For more information, see the Javadoc for that interface in the `<pf_install>/pingfederate/sdk/doc` directory.

PingFederate displays the language indicated in the cookie if the language is supported in the `language-packs` directory. If the matching localization file is not found, PingFederate defaults to the browser settings.

Retrieval of localized messages

Retrieval of localized messages is supported through the `LanguagePackMessages` class available in the PingFederate SDK. An instance of this class is passed into every template file and made available for use there. For more information, see the Javadoc for the class in the `<pf_install>/pingfederate/sdk/doc` directory.

Configuring a password policy

PingFederate applies a configurable policy to passwords, pass phrases, and shared secrets defined by administrators in the administrative console.

About this task

These fields include, but are not limited to:

- Passwords used by HTTP Basic authentication for:
  - Inbound SOAP messages from partners via back-channel calls
  - WS-Trust STS
- Shared secrets used by the credentials defined for:
  - Attribute Query
  - Java Management Extensions (JMX)
  - Connection Management
  - Single sign-on (SSO) Directory Service
- Passwords used by instances of the Simple Username Password Credential Validator (PCV)
- Passwords used for encrypting certificates exported with their private keys
- Pass phrases used by identity provider (IdP) Discovery
- Passwords used by administrative console credentials when native authentication is used

> **Note**
>
> Passwords external to PingFederate, such as passwords used by instances of the datastores, are not subject to this password policy.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/password-rules.xml` file.
2. Save the changes.
3. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node. You do not have to change or restart PingFederate on the engine nodes.

Managing cipher suites

You can enable, disable, and re-order cipher suites in PingFederate.

About this task

The SSL/TLS server-client handshake involves negotiating cipher suites to use for encryption and decryption on each side of a secured transaction. You can find cipher suites in the following configuration files:

- `com.pingidentity.crypto.SunJCEManager.xml`
- `com.pingidentity.crypto.AWSCloudHSMJCEManager.xml`

- `com.pingidentity.crypto.LunaJCEManager.xml`
- `com.pingidentity.crypto.NcipherJCEManager.xml`
- `com.pingidentity.crypto.BCFIPSJCEManager.xml`

These cipher-suite configuration files are located in the `<pf_install>/server/default/data/config-store` directory. These files comment out weaker cipher suites. To ensure the most secure transactions, retain this cipher-suite configuration.

> **Important**
>
> Because of the import restrictions of some countries, Oracle Server Java SE Runtime Environment (JRE) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, enable the Java Cryptography Extension (JCE) unlimited strength jurisdiction policy. For more information, see the Java 8 release notes in Oracle's documentation.
> For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide in Oracle's documentation.

Starting with PingFederate 9.1, cipher suites are selected based on the order that they are listed in the cipher-suite configuration file for new installations. For upgrades, you can enable the same selection mechanism as well.

Steps

For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

To enable cipher-suite selection based on listing order after an upgrade, follow these steps.

1. Create a new text file with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<c:config xmlns:c="http://www.sourceid.org/2004/05/config">
    <c:item name="prefer-server-cipher-suites">true</c:item>
</c:config>
```

2. Save this file as `cipher-suite-settings.xml` in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

3. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

   > **Important**
   >
   > For each engine node, restart PingFederate to load the changes made in the `cipher-suite-settings.xml` file after the configuration is replicated.

Related links

- Secure sockets layer

Manage externally stored authentication sessions

Authentication sessions control when previously authenticated users are redirected back to the authentication sources on subsequent requests for browser-based single sign-on (SSO) and PingFederate user-facing applications.

When you enable authentication sessions, PingFederate maintains session data in memory. PingFederate also supports maintaining session data both in memory and on an external storage. This optional capability allows your organization to support use cases where a longer session duration or a greater resilience against restarts of PingFederate and browsers is desired.

PingFederate supports storing persistent authentication sessions on a database server or a PingDirectory server. When stored on a database server, the default cleanup task removes expired authentication sessions once a day. If stored on a PingDirectory server, configure a cleanup plugin in PingDirectory to suit the needs of your organization.

Related links

- Sessions
- Defining a datastore for persistent authentication sessions

Managing authentication sessions stored in the database

PingFederate uses a cleanup task to remove expired authentication sessions from the configured database once a day. The cleanup task determines whether a session can be removed by looking at the session's expiration timestamp and the current time.

About this task

Any session that has an expiration timestamp older than the current time by a configurable offset is subject to removal. As needed, the cleanup task can look at the session's last activity timestamp instead. The cleanup task removes 500 expired sessions at a time until all expired sessions are removed. If expired sessions are growing rapidly, you can optionally increase the frequency of the cleanup task.

> **Note**
>
> Increasing the frequency of the cleanup task or the number of expired sessions to be removed per batch (or both) adds more workload to your storage server. Make changes gradually to observe the impact.

> **Important**
>
> In a clustered PingFederate environment, the cleanup task runs only on the console node. If adjustments are required, make them on the console node. No changes are required on any of the engine nodes.

Steps

1. **Optional:** Adjust the frequency of the cleanup task.
   1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/timer-intervals.xml` file.
   2. Update the `StoredSessionCleanerInterval` value, in milliseconds.

      The default value is `86400000`, which is 24 hours.
   3. Save your changes.

2. **Optional:** Configure other cleanup options.

    1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl.xml` file.

        See the following table for more information about each field.

Field	Description
ExpiredSessionGroupBatchSize	The number of expired authentication sessions to be removed per batch. The default value is `500`.
ExpirationTimeColumnName	The column of which its value determines whether an authentication session has expired in the context of the cleanup task. Valid options are `expiry_time` and `last_activity_time`. **`expiry_time`** Set to `expiry_time` if the cleanup task should only remove persistent authentication sessions that have expired.The cleanup task determines if a session can be removed by looking at the session's expiration timestamp and the current time. If the expiration timestamp is older than the current time by the number of minutes specified by the `ExpirationTimeOffsetMins` field, the session is subject to removal. **`last_activity_time`** Set to `last_activity_time` if the clean task should remove persistent authentication sessions that have been left idle.The cleanup task determines if a session can be removed by looking at the session's last activity timestamp and the current time. If the last activity timestamp is older than the current time by the number of minutes specified by the `ExpirationTimeOffsetMins` field, the session is subject to removal.For example, if PingFederate should remove persistent authentication sessions for which the last activity time is more than three weeks ago, set the `ExpirationTimeColumnName` value to `last_activity_time` and the `ExpirationTimeOffsetMins` value to `30240`. The default value is `expiry_time`.
ExpirationTimeOffsetMins	The offset, in minutes, relative to the current time. The default value is `10`.

    2. Save your changes.

3. If you have made any changes, restart PingFederate.

    In a clustered PingFederate environment, you do not have to change or restart PingFederate on any of the engine nodes.

Related links

- Sessions
- Defining a datastore for persistent authentication sessions

Managing authentication sessions stored in PingDirectory

When storing persistent authentication sessions on a PingDirectory server, you must also configure a cleanup plugin in PingDirectory to remove expired authentication sessions from your directory server.

Steps

1. Disable the PingFederate cleanup task.

   > **Important**
   >
   > For a clustered PingFederate environment, make these changes on the console node. None of the engine nodes require any changes.

   1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/timer-intervals.xml` file.
   2. Update the `StoredSessionCleanerInterval` value to `0`.
   3. Save your changes.
   4. Restart PingFederate.
2. Sign on to the PingDirectory administrative console.
3. Go to **Configuration > Plugin Root**.
4. On the **Plugin Root** window, click **New Plugin**, and then select **Purge Expired Data Plugin**.
5. Configure a new instance of the **Purge Expired Data Plugin**.

   See the following table for information about each required field.

Field	Description
Name	The name of this plugin instance.
Enabled	The status of this plugin instance. Select the check box to enable this plugin instance. Clear the check box to disable this plugin instance. This check box is not selected by default.

Field	Description
Datetime Attribute	The attribute value determines whether an authentication session has expired in the context of this plugin instance. Valid options are `pf-authn-session-group-expiry-time` and `pf-authn-session-group-last-activity-time`. **`pf-authn-session-group-expiry-time`** Set to `pf-authn-session-group-expiry-time` if this plugin instance should only remove persistent authentication sessions that have expired.This plugin instance determines if a session can be removed by looking at the session's expiration timestamp and the current time. If the expiration timestamp is older than the current time by the number of minutes specified by the **Expiration Offset** field, the session is subject to removal. **`pf-authn-session-group-last-activity-time`** Set to `pf-authn-session-group-last-activity-time` if the clean task should remove persistent authentication sessions that have been left idle.This plugin instance determines if a session can be removed by looking at the session's last activity timestamp and the current time. If the last activity timestamp is older than the current time by the number of minutes specified by the **Expiration Offset** field, the session is subject to removal.For example, if PingFederate should remove persistent authentication sessions for which the last activity time is more than three weeks ago, set the **Datetime Attribute** value to `pf-authn-session-group-last-activity-time` and the **Expiration Offset** value to `3 w`.
Datetime Format	The format of the attribute specified in the **Datetime Attribute** field. Select **generalized-time** from the list. The default selection is **generalized-time**.
Expiration Offset	The offset relative to the current time. Enter an integer to indicate the time value, followed by its unit of measurement. This field has no default value.
Purge Behavior	The method how this plugin instance removes expired data. Select **subtree-delete-entries** from the list. This field has no default selection.
Polling Interval	The frequency of which this plugin instance should be run. Enter an integer to indicate the time value, followed by its unit of measurement. This field has no default value.
Max Updates Per Second	This setting smooths out the performance impact on the server by throttling the purging to the specified maximum number of updates per second. To avoid a large backlog, this value should be set comfortably above the average rate that expired data is generated. When you select **subtree-delete-entries** from the **Purge Behavior** list, deletion of the entire subtree is considered a single update for the purposes of throttling. This field has no default value.

6. Click **Save**.

Related links

- Sessions
- Defining a datastore for persistent authentication sessions

OAuth persistent grants cleanup

PingFederate provides two cleanup tasks for persistent grants. One task manages expired grants, while another task caps the number of grants based on a combination of user, client, grant type, and authentication context.

Persistent authorizations include those obtained by OAuth clients in the following ways:

- Grants obtained or updated using the authorization code, resource owner credentials, or device authorization grant type, in conjunction with the refresh token grant type

> **Note**
>
> If the use cases involve mapping attributes from authentication sources, such as IdP adapter instances or IdP connections, or password credential validator (PCV) instances to the access tokens, directly or through persistent grant-extended attributes, storing these attributes from authentication sources and their values along with the persistent grants maintains them for reuse when clients subsequently present refresh tokens for new access tokens.

- Grants obtained or updated by using the implicit grant type, for which PingFederate is configured to reuse existing persistent grants

> **Note**
>
> If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens, runtime procedures obtain attribute values for each token request, but persistent grants do not store with attributes or their values.

Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

> **Note**
>
> PingFederate does not factor in the **Persistent Grant Idle Timeout** setting during grant cleanup. Ensure the grant datastore has the disk space needed to store expired grants because they exceeded the **Persistent Grant Idle Timeout** setting.

Managing expired persistent grants

PingFederate removes expired persistent grants once a day. The cleanup task removes 500 expired grants at a time until all expired grants are removed.

About this task

If expired grants are growing rapidly, you can optionally increase the frequency of the cleanup task.

> **Note**
>
> Increasing the frequency of the cleanup task or the number of expired sessions to be removed per batch adds more workload to your storage server. Make gradual changes, if any, to observe the impact.

> **Important**
>
> In a clustered PingFederate environment, the cleanup task runs only on the console node. If adjustments are required, make them on the console node. No changes are required on any of the engine nodes.

When storing persistent grants on a PingDirectory server that is version 7.0 or later, you can use the PingFederate cleanup task or configure a cleanup plugin in PingDirectory instead. The plugin allows fine-grained control over various aspects of the cleanup task, which might improve the performance impact. For more information and configuration steps, see Managing expired persistent grants in PingDirectory.

Steps

1. **Optional:** Adjust the frequency of the cleanup task.
   1. Edit the `timer-intervals.xml` `<pf_install>/pingfederate/server/default/data/config-store` directory.
   2. Update the `AccessGrantCleanerInterval` value, in milliseconds.

      The default value is `86400000`, which is 24 hours.
   3. Save your changes.
2. **Optional:** Adjust the number of expired grants to be removed per batch.
   1. Edit the configuration file relevant to your storage platform.

      This configuration file is located in the `<pf_install>/pingfederate/server/default/data/config-store` directory, as described in the following table.

Storage platform	Configuration file
Database server	`org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml`
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl.xml`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl.xml`
Oracle Unified Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl.xml`
   2. Update the `ExpiredGrantBatchSize` value.

      The following example shows an updated value of 400.

```
 file, located in the<?xml version="1.0" encoding="UTF-8"?>
<c:config xmlns:c="http://www.sourceid.org/2004/05/config">
    ...
    <c:item name="ExpiredGrantBatchSize">400</c:item>
    ...
</c:config>
```

The default value is `500`.

    3. Save your changes.

3. After you have made changes, restart PingFederate.

   In a clustered PingFederate environment, you do not have to change or restart PingFederate on any of the engine nodes.

Related links

- Grant types
- Grant storage and management

Managing expired persistent grants in PingDirectory

When storing OAuth persistent grants on a PingDirectory server that is version 7.0 or later, you can configure a cleanup plugin in PingDirectory to remove expired data from your directory server.

About this task

This PingDirectory plugin allows fine-grained control over various aspects of the cleanup task. For example, you can configure the maximum number of updates per second to improve the performance impact.

Steps

1. Disable the PingFederate cleanup task.

   > **Important**
   >
   > For a clustered PingFederate environment, make these change on the console node. No changes are required on any of the engine nodes.

   1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/timer-intervals.xml` file.
   2. Update the `AccessGrantCleanerInterval` value to `0`.
   3. Save your changes.
   4. Restart PingFederate.
2. Configure an instance of the PingDirectory plugin to clean up expired data.
   1. Sign on to the PingDirectory administrative console.
   2. Go to **Configuration > Plugin Root**.

3. Click **New Plugin** and then select **Clean up Expired PingFederate Persistent Access Grants Plugin**.
4. Configure a new instance of the **Clean up Expired PingFederate Persistent Access Grants Plugin**.

   See the following table for information about each required field.

Field	Description
Name	The name of this plugin instance.
Enabled	The status of this plugin instance. Select the check box to enable this plugin instance. Clear the check box to disable this plugin instance. This check box is not selected by default.
Base DN	The distinguished name (DN) that points to the access grants location. For more information, see the inline comment and the `access-grant-ldap-pingdirectory.ldif` file in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.
Polling Interval	The frequency of which this plugin instance should be run. Enter an integer to indicate the time value, followed by its unit of measurement. The default value is `5 m`.
Max Updates Per Second	This setting smooths out the performance impact on the server by throttling the purging to the specified maximum number of updates per second. To avoid a large backlog, this value should be set above the average rate that expired data is generated. The default value is `100`.

5. Click **Save**.

Related links

- Grant types
- Grant storage and management
- Indexing grant attributes in PingDirectory

Managing cleanup of persistent grants

PingFederate is capable of capping the number of persistent grants based on a combination of user, client, grant type, and authentication context.

About this task

Capping the number of persistent grants helps limit the data stored for persistent grants, especially in scenarios where clients frequently request authorization in a single context.

When PingFederate needs to record a new grant, it checks whether such creation will push the number of grants beyond the limit. If it does, PingFederate creates the grant and then removes just enough grants so that the number of grants is capped at the limit. This cleanup task starts from the oldest grant, expired or not, and continues forward if it needs to remove multiple grants. For performance reasons, this cleanup task also limits the number of grants it can remove per attempt. If it cannot remove all grants in excess of the limit, it removes what it can and repeats the process when PingFederate needs to record a new grant.

This cleanup runs on every engine node in a clustered PingFederate environment. Also, it does not replace the cleanup task or the PingDirectory plugin engineered to manage expired grants. Working together, they keep the size of the grant datastore under control.

The default limit is 100 grants per user, client, grant type, and authentication context. Depending on the storage platform, the default maximum number of grants that this cleanup task can remove per attempt varies.

This cleanup task is enabled on new installations. When upgrading from version 9.1 or an earlier version, it is disabled. You can enable it by editing an XML configuration file.

Steps

1. Edit the configuration file relevant to your storage platform.

   This configuration file is located in the `<pf_install>/pingfederate/server/default/data/config-store` directory, as described in the following table.

Storage platform	Configuration file
Database server	`org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml`
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl.xml`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl.xml`
Oracle Unified Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl.xml`

2. Locate for the following comments.

```
...
<!--
    Maximum number of persistent grants allowed to store in the database per user,
    client and grant type and authentication context qualifier.

    Setting this to a value <= 0 will turn this limit off
    Default configuration:
    <c:item name="maxPersistentGrants">100</c:item>
-->
 <c:item name="maxPersistentGrants">100</c:item>
<!--
    Maximum number of persistent grants to delete when max allowed is reached
    during new grant creation.

    Setting this to a value <= 0 will turn this limit off
    Default configuration:
    <c:item name="maxPersistentGrantsToRemoveBatchSize">n</c:item>
-->
 <c:item name="maxPersistentGrantsToRemoveBatchSize">n</c:item>
...
```

The `maxPersistentGrants` value represents the maximum number of grants based on a combination of user, client, grant type, and authentication context.

The `maxPersistentGrantsToRemoveBatchSize` value represents the maximum number of grants that the cleanup task would remove per attempt. Its default value (*n*) varies depending on the storage platform, `50` for a database server and `10` for a directory server.

> **Note**
>
> The `maxPersistentGrants` and `maxPersistentGrantsToRemoveBatchSize` items exist only on new installations starting with version 9.2. When upgrading from version 9.1 or an earlier version, the upgrade tools only insert the comments for reference.

3. **Optional:** Adjust the `maxPersistentGrants` and `maxPersistentGrantsToRemoveBatchSize` values.

   Use integers only.

4. To enable this cleanup task after upgrading from version 9.1 or an earlier version, insert the `maxPersistentGrants` and `maxPersistentGrantsToRemoveBatchSize` items into the configuration file.

   You can use the default values based on the inline comment. You can also adjust the values to suit the needs of your organization.

5. Save your changes.

6. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

Related links

- Grant types

- Grant storage and management

Specifying the domain of the PF cookie

PingFederate identifies sessions by their respective PingFederate cookie. You can specify the domain of these cookies.

About this task

By default, the PingFederate cookie is set without domain information in the HTTP header.

```
Set-Cookie: PF=zOv4xxmzDI2rx1TFBFy78X;Path=/;Secure;HttpOnly
```

You can configure PingFederate to return the `Set-Cookie` HTTP header with domain information as needed.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/session-cookie-config.xml` file.
2. Modify the `cookie-domain` element.

   Example:

   ```
   <c:item name="cookie-domain">.example.com</c:item>
   ```
3. Save the change.
4. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node. Then, click **Replicate Configuration** on **System > Server > Cluster Management**.

Result

After you activate this change, PingFederate includes domain information in its `Set-Cookie` HTTP header.

```
Set-Cookie: PF=aDfPx6uwbbWGFhwE6zEhEG;Path=/;Domain=.example.com;Secure;HttpOnly
```

Specifying the domain of the PF.PERSISTENT cookie

PingFederate identifies persistent authentication sessions by their respective PF.PERSISTENT cookie. You can specify the domain of this cookie.

About this task

By default, the PF.PERSISTENT cookie is set without domain information in the HTTP header.

```
Set-Cookie: PF.PERSISTENT=UoBlPlf16V2oYAEPot2DnpUOXxitK7au;Path=/;Expires=Sat, 06-Nov-2021 00:48:08 GMT;Max-Age=94608000;Secure;HttpOnly
```

You can configure PingFederate to return the Set-Cookie HTTP header with domain information, as needed.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/persistent-session-cookie-config.xml` file.

2. Modify the `cookie-domain` element.

   Example:

   `<c:item name="cookie-domain">.example.com</c:item>`

3. Save the change.

4. Restart PingFederate.

5. If you’re running PingFederate in a clustered environment, perform the preceding steps on the console node. Then go to **System > Server > Cluster Management** and click **Replicate Configuration**.

Result

After you activate this change, PingFederate includes domain information in its Set-Cookie HTTP header.

```
Set-Cookie: PF.PERSISTENT=tOYwPM7VFMeluUyeu0EKQLL0DCJyVOqG;Path=/;Domain=.example.com;Expires=Sat, 06-Nov-2021 01:00:34 GMT;Max-Age=94608000;Secure;HttpOnly
```

Related links

- Sessions

Extending the lifetime of the PingFederate cookie

PingFederate identifies sessions by their respective PingFederate cookies. You can manually extend the lifetime of these cookies.

About this task

Some adapters, such as the HTML Form Adapter, also utilize the PingFederate cookie to manage their adapter-sessions. The PingFederate cookie is a session cookie by default. You can extend the lifetime of the PingFederate cookie by making it a persistent cookie. Unlike session cookies, persistent cookies are saved to disk, enabling the browser to reuse them when restarted.

> **Tip**
>
> Alternatively, you can configure PingFederate to store authentication sessions externally and leverage them as users request protected resources after restarting their browsers. For more information, see Sessions.

Steps

1. Edit the `session-cookie-config.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

2. Modify the `cookie-max-age` value.

   The default value, `-1`, makes the PingFederate cookie a session cookie. A positive integer defines the age of the persistent cookie in seconds.

3. Save the change.

4. Restart PingFederate.

For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

Configuring forward proxy server settings

You can configure PingFederate to send web traffic, such as HTTP and HTTPS, so that it initiates through a forward proxy server.

Steps

1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
2. Locate for the following properties:

   ```
   #http.proxyHost=<HTTP_PROXY_HOST>
   #http.proxyPort=<HTTP_PROXY_PORT>
   #http.proxyUser=<HTTP_PROXY_USER>
   #http.proxyPassword=<HTTP_PROXY_PASSWORD>
   #https.proxyHost=<HTTPS_PROXY_HOST>
   #https.proxyPort=<HTTPS_PROXY_PORT>
   #https.proxyUser=<HTTPS_PROXY_USER>
   #https.proxyPassword=<HTTPS_PROXY_PASSWORD>
   #http.nonProxyHosts=*.internal.com|localhost
   ```

3. **Optional:** Configure forward proxy server settings for HTTP traffic:
   1. Remove the number sign (`#`) in front of `http.proxyHost` and `http.proxyPort`.
   2. Enter the host name or the IP address of the forward proxy server.
   3. To specify a username and password that are required to connect to the forward proxy server, uncomment `http.proxyUser` and `http.proxyPassword`.
   4. Enter the username and password credentials.
4. **Optional:** Configure forward proxy server settings for HTTPS traffic:
   1. Remove the number sign in front of `https.proxyHost` and `https.proxyPort`.
   2. Enter the host name or the IP address of the forward proxy server.
   3. To specify a username and password that are required to connect to the forward proxy server, uncomment `https.proxyUser` and `https.proxyPassword`.
   4. Enter the username and password credentials.

      > **Note**
      >
      > The `http.proxyUser` and `https.proxyUser` settings both support obfuscation and secret manager references. For more information, see Secret managers.

> **Tip**
>
> You can obfuscate the password used to access the forward proxy server by running the obfuscate utility, located in the `<pf_install>/pingfederate/bin` directory:
>
> - `obfuscate.bat` for Windows
> - `obfuscate.sh` for Linux
>
> Use the actual password as an argument and copy the entire result into the value for the password parameter in `run.properties`.

5. **Optional:** Configure an exclusion list:

    1. Remove the number sign in front of `http.nonProxyHosts`.

    2. Specify one or more destinations where PingFederate is not required to proxy its HTTP and HTTPS traffic through the forward proxy server.

    This property supports multiple values separated by the pipe character (`\|`) and the wildcard character (`#`) for pattern matching.
    `*.example.com\|localhost`

6. **Optional:** If you want to enable basic authentication for an HTTP or HTTPS target site, you can remove `Basic` from `jdk.http.auth.proxying.disabledSchemes=Basic` or `jdk.http.auth.tunneling.disabledSchemes=Basic`, respectively.

    > **Important**
    >
    > You should only use digest authentication with proxy servers. Basic authentication is not recommended because the proxy credentials are transmitted to the server without encryption.

7. Save your changes.

8. Restart PingFederate.

    For a clustered PingFederate environment, repeat these steps on each node.

Adding custom HTTP response headers

The PingFederate administrative console and runtime server are capable of returning custom HTTP response headers, such as HTTP Strict-Transport-Security (HSTS), to enforce HTTPS-based access and P3P.

Steps

1. Edit the `response-header-admin-config.xml` file or the `response-header-runtime-config.xml` file, or both, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

2. Save your changes.

3. Restart PingFederate.

    For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

Configuring validation for the AudienceRestriction element

You can configure validation for the `AudienceRestriction` value in a SAML response.

About this task

For any identity provider (IdP) connection configured with multiple virtual server IDs, the `AudienceRestriction` value in a SAML response must match the virtual server ID information embedded in the protocol endpoint at which PingFederate receives the message.

You can disregard this validation condition on a per-connection basis.

Steps

1. Edit the `org.sourceid.saml20.util.VirtualIdentityUtil.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

2. Optionally, if you want to disregard the validation condition for an IdP connection, add its **Partner's Entity ID** value as an entry inside the `c:map` element.

   Example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <c:config xmlns:c="http://www.sourceid.org/2004/05/config">
       <c:map name="AllowAnyVirtualServerIdInAudience">
            <c:item name="www.example.com"/>
            <c:item name="www.example.org"/>     </c:map>
   </c:config>
   ```

   Result:

   In this example, the first entry adds the IdP connection with a **Partner's Entity ID** of `www.example.com` to the list. This is so that PingFederate no longer returns an error if the `AudienceRestriction` value in a SAML response does not match the virtual server ID information embedded in the protocol endpoint at which PingFederate receives the message. The second entry has the same effect for the IdP connection with a **Partner's Entity ID** of `www.example.org`.

3. Save your changes.

4. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

Customizing a configuration endpoint response

About this task

You can customize the PingFederate `openid-configuration.template.json` template to configure both OpenID Connect (OIDC) discovery and OAuth authorization server metadata endpoints. To suit multiple use cases simultaneously, you can customize the amount of configuration information and add conditional statements to return different responses, based on information from the requests.

Customizing the OpenID Provider configuration endpoint response

The OpenID Provider (OP) configuration endpoint at `/.well-known/openid-configuration` provides configuration information for the OAuth clients to interface with PingFederate using the OIDC protocol.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/template/openid-configuration.template.json` file to specify the desired information to be returned by the OAuth metadata configuration endpoint.

   Multiple samples are provided, including sample statements using the `$HttpServletRequest` and `$HttpServletResponse` objects to get and set values.

2. Save your changes.

   Template customization doesn't require a restart of PingFederate.

   For a clustered PingFederate environment, repeat these steps on each node.

Customizing the OAuth authorization server metadata endpoint response

The OAuth authorization server metadata endpoint at `/.well-known/oauth-authorization-server` provides configuration information for the OAuth clients to interface with PingFederate using the OAuth 2.0 protocol.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/template/openid-configuration.template.json` file to specify the desired information to be returned by the OAuth metadata configuration endpoint.

   Multiple samples are provided, including sample statements using the `$HttpServletRequest` and `$HttpServletResponse` objects to get and set values.

2. Save your changes.

   Template customization doesn't require a restart of PingFederate.

   For a clustered PingFederate environment, repeat these steps on each node.

Related links

- OpenID Provider configuration endpoint
- OAuth authorization server metadata endpoint
- HttpServletRequest in the Java EE 8 Specification APIs documentation.
- HttpServletResponse in the Java EE 8 Specification APIs documentation.

Customizing the heartbeat message

The heartbeat endpoint, `/pf/heartbeat.ping`, returns a customizable `OK` browser message and an HTTP 200 status indication if the PingFederate server is running.

About this task

You can customize the message by modifying the `pf.heartbeat.system.monitoring` PingFederate property.

> **Note**
>
> If a GET request receives a connection error or an HTTP status code other than 200, the server associated with the endpoint is down or malfunctioning.

Steps

1. Set the `pf.heartbeat.system.monitoring` property in the `<pf_install>/pingfederate/bin/run.properties` file to `true` or `false`.

   When `pf.heartbeat.system.monitoring` is set to `false`, the `/pf/heartbeat.ping` endpoint returns `OK`. When set to `true`, the `/pf/heartbeat.ping` endpoint returns all available statistics.

   > **Caution**
   >
   > Before exposing these additional statistics on the heartbeat endpoint, ensure the endpoint cannot be reached beyond the load balancer so that this information isn't publicly available.

2. Restart PingFederate.

3. If you want to customize the information returned by the heartbeat endpoint, edit the `heartbeat.page.template` file, located in the `<pf_install>/pingfederate/server/default/conf/template` directory.

   > **Note**
   >
   > Metrics can make the heartbeat response very large, especially if your PingFederate configuration has many connections or adapters. You can use wildcard patterns in `heartbeat.page.template` to limit the metrics included in the response.

   Template customization does not require a restart of PingFederate. For a clustered PingFederate environment, repeat these steps on each node.

4. If you want to specify percentiles in addition to, or in place of, the default 90th percentile in the statistics reported on the heartbeat:

   1. Edit the `com.pingidentity.monitoring.MonitoringService.xml` file located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   2. Change the value of the `StatisticsPercentilesList` item to the preferred percentile. You can enter a single value, such as `99.9`, or multiple values separated by commas, such as `80,90,99.5`.

      > **Note**
      >
      > The `StatisticsPercentilesList` item allows you to customize the percentiles displayed in the heartbeat endpoint response of the metrics ending in `.<StatisticsPercentilesList>.percentile`. Percentiles can be a helpful way to understand a metric's distribution and identify patterns or trends over time. They can also be used to set performance targets or to identify bottlenecks in a system. In the context of server response metrics, you can use percentiles to compare a server's response time to other servers or previous periods. For more information, including a complete list of server metrics and their descriptions, see Liveliness and responsiveness.

      Example:

Setting the `StatisticsPercentilesList` item to `50` will display the 50th percentile of a server's response time in the heartbeat endpoint response. A value of `200` milliseconds for the 50th percentile means that 50% of the server's responses were faster than 200 milliseconds, while 50% were slower. Similarly, a value of `500` milliseconds for the 95th percentile means that 95% of the server's responses were faster than 500 milliseconds, while 5% were slower.

3. Save your changes.

5. If you want to completely disable tracking of adapter or connection metrics:

    1. Edit the `com.pingidentity.monitoring.MonitoringService.xml` file located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
    2. To disable adapter metrics, change the value of the `EnableAdapterMetrics` item to `false`.
    3. To disable connection metrics, change the value of the `EnableConnectionMetrics` item to `false`.
    4. Save your changes.

Customizing the favicon for application and protocol endpoints

PingFederate provides a favorite icon (favicon) for its application, such as `/idp/startSSO.ping`, and protocol endpoints, such as `/idp/SSO.saml2`.

Steps

1. In the `<pf_install>/pingfederate/server/default/conf/template/assets/images` directory, replace the `favicon.ico` file.
2. Restart PingFederate.

For a clustered PingFederate environment, repeat these steps on each node.

Configuring the behavior of searching multiple datastores with one mapping

If a datastore uses results from previous queries as input, and if the previous queries return no result, PingFederate records a warning message in the server log and continues with the request by querying the next datastore in the attribute source setup.

About this task

This default behavior applies to all lookup configurations using multiple datastores in one mapping. For more information, see Attribute mapping with multiple data sources.

If you prefer PingFederate to abort the request immediately, which is the default behavior of many earlier versions of PingFederate, you can override the behavior by modifying a configuration file. Like the default behavior, this override also applies to all lookup configurations using multiple datastores in one mapping.

Steps

1. Edit the `org.sourceid.saml20.domain.AttributeMapping.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

> **Note**
>
> If this file does not exist, you must create it.

2. To override the default behavior, change the value of the `AbortOnAttrLookupFailure` element from `false`, the default value, to `true`.

   The following is an example of a modified `org.sourceid.saml20.domain.AttributeMapping.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<c:config xmlns:c="http://www.sourceid.org/2004/05/config">
    <c:item name="AbortOnAttrLookupFailure">true</c:item>
</c:config>
```

> **Note**
>
> Removing the `org.sourceid.saml20.domain.AttributeMapping.xml` file from the `<pf_install>/pingfederate/server/default/data/config-store` directory also has the same effect as setting the value of the `AbortOnAttrLookupFailure` element to `true`.

   For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on **System > Server > Cluster Management**.

Example

Expected result when this override is set

If a datastore uses results from previous queries as input, and if the previous queries return no result, PingFederate records an error message in the server log, aborts the request immediately, and returns an error message to the user, the application, or the partner.

Signing algorithms

PingFederate supports several signing algorithms.

You can use signing algorithms to digitally sign tokens, messages, and files. The following table lists the supported signing algorithms and their identifying URIs.

Algorithm name	Algorithm URI
DSA SHA1	http://www.w3.org/2000/09/xmldsig#dsa-sha1
RSA SHA1	http://www.w3.org/2000/09/xmldsig#rsa-sha1
RSA SHA256	http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
RSA SHA384	http://www.w3.org/2001/04/xmldsig-more#rsa-sha384
RSA SHA512	http://www.w3.org/2001/04/xmldsig-more#rsa-sha512

Algorithm name	Algorithm URI
RSASSA-PSS SHA256	http://www.w3.org/2007/05/xmldsig-more#sha256-rsa-MGF1
RSASSA-PSS SHA384	http://www.w3.org/2007/05/xmldsig-more#sha384-rsa-MGF1
RSASSA-PSS SHA512	http://www.w3.org/2007/05/xmldsig-more#sha512-rsa-MGF1
ECDSA SHA256	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256
ECDSA SHA384	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384
ECDSA SHA512	http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512

This table shows the URIs of the signing algorithms that PingFederate supports.

System settings

The **System** menu provides access to system-related settings.

Depending on the setup of PingFederate, menu items vary. Menu items are organized into four groups:

- **Server**
- **Metadata**
- **Monitoring & Notifications**
- **External Systems**

Server

You can configure various PingFederate settings.

On the **Extended Properties** window, you can configure protocol settings and local administrative accounts, import your license file, export or import a configuration archive, review other runtime servers in the same cluster and replicate configuration from the administrative server to the clustered runtime servers. You can also configure virtual host names, and configure extended properties for connections and OAuth clients. See the following topics for more information:

- Administrative accounts
- License management
- Configuration archive
- Cluster management
- Virtual host names
- Extended properties

Protocol settings

On the **Protocol Settings** window, you can configure the base URL of your PingFederate environment, and settings for WS-Trust STS authentication, outbound provisioning, and IdP discovery.

Specifying federation information

Federation information identifies your federation deployment to your partners, according to the protocols you support.

About this task

You must provide an ID that uniquely identifies your federation gateway for each protocol you support. For WS-Trust security token service (STS), IDs are required for both SAML 2.0 and SAML 1.x, regardless of browser-based single sign-on (SSO) protocol support or the type of token expected to be issued, to ensure that the STS will perform correctly under all conditions.

> **Note**
>
> Each ID normally applies across all connection partners for a given protocol. However, if your implementation requires different IDs for the same protocol, you can use virtual server IDs. For more information, see Federation planning checklist.

Steps

1. Go to **System > Server** to open the **Protocol Settings** window.
2. On the **Federation Info** tab, provide the required information.

   For more information, see the following table.

Field	Description
Base URL	The fully qualified host name, port, and path (if applicable) on which the PingFederate server runs. This field is used to populate configuration settings in metadata files. For more information, see Metadata export.
SAML 2.0 Entity ID	This ID defines your organization as the entity operating the server for SAML 2.0 transactions. It is usually defined as an organization's URL or a DNS address, for example: `pingidentity.com`. The SAML SourceID used for artifact resolution is derived from this ID using SHA1.
SAML 1.x Issuer/Audience	This ID identifies your federation server for SAML 1.x transactions. As with SAML 2.0, it is usually defined as an organization's URL or a DNS address. The SourceID used for artifact resolution is derived from this ID using SHA1.
SAML 1.x Source ID	(Optional) If supplied, the Source ID value entered here is used for SAML 1.x, instead of being derived from the SAML 1.x Issuer/Audience.
WS-Federation Realm	The URI of the realm associated with the PingFederate server. A realm represents a single unit of security administration or trust.

3. Click **Next** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring WS-Trust settings

You can configure PingFederate to require that client applications provide credentials to access the security token service (STS).

About this task

While this is an optional configuration, it is recommended for identity provider (IdP) configurations using the Username Token Processor. For other token processors and token generators, trust in the identity of the client is conveyed within the token itself and verified as part of processing. However, you can still configure authentication requirements to add another layer of security by limiting access to only authenticated clients.

> **Note**
>
> You can configure STS authentication to either apply globally to all token formats and for all IdP and service provider (SP) partner connections, or token-to-token mappings, using more fine-grained controls at the connection level through issuance criteria.

> **Note**
>
> WS-Trust STS settings can also be configured through the PingFederate Administrative API platform. For more information, see Accessing the API interactive documentation.

Steps

1. Go to **System > Server** to open the **Protocol Settings** window
2. On the **WS-Trust STS Settings** tab, click **Configure WS-Trust STS Authentication**.

   Follow the configuration wizard to complete the task. For more information, see Configuring STS authentication
3. Click **Next** and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring outbound provisioning settings

You can select the database that PingFederate should use internally to facilitate provisioning for service providers when PingFederate is configured as an identity provider (IdP).

Before you begin

Before configuring outbound provisioning settings, you must enable outbound provisioning through the `pf.provisioner.mode` property in the `<pf_install>/pingfederate/bin/run.properties` file. For more information, see Configuring PingFederate properties.

If you want to use failover provisioning, configure the `provisioner.node.id` and `provisioner.failover.grace.period` properties, which are also located in `<pf_install>/pingfederate/bin/run.properties`. These properties are described in Deploying provisioning failover.

About this task

The database stores the state of synchronization between the source datastore and the target datastore, enabling periodic checking to determine whether updates are required at the target site. PingFederate checks the source datastore for changes every minute by default. As needed, you can change the provisioning synchronization frequency on this tab as well.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

> **Note**
>
> PingFederate is tested with Amazon Aurora (MySQL and PostgreSQL), Microsoft SQL Server, Oracle Database, Oracle MySQL, and PostgreSQL as internal provisioning datastores. A demonstration-only, embedded HSQLDB database is installed by default. Scripts to aid setup are in the directory `<pf_install>/pingfederate/server/default/conf/provisioner/sql-scripts`.

Steps

1. Go to **System** > **Server** > **Protocol Settings**.
2. On the **Outbound Provisioning** tab, from the **Provisioning Data Store** list, select a datastore.

   If the datastore you want is not shown in the list, PingFederate is not yet configured to access the store. Click **Manage Data Stores** to create a connection to the datastore.
3. Change the **Synchronization Frequency** value.

   The default value is `60` seconds.
4. Click **Next** and continue with the rest of the configuration.

   > **Tip**
   >
   > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring standard IdP Discovery

SAML 2.0 identity provider (IdP) Discovery provides a cookie-based look-up mechanism to identify a user's IdP dynamically during a service provider (SP)-initiated single sign-on (SSO) event when the IdP is not otherwise specified.

About this task

This mechanism can be helpful in cases where an SP might be a hub for several IdPs in an identity federation.

In addition to supporting SAML 2.0 IdP Discovery, PingFederate provides a cross-protocol, proprietary mechanism allowing a PingFederate SP server to write a persistent browser cookie. The cookie contains a reference to the IdP partner with whom the user previously authenticated for SSO. For more information, see Configuring IdP discovery using a persistent cookie.

> **Tip**
>
> An SP can also include the discovery mechanism within the application. For instance, an SP can provide vanity URLs to isolate one set of end users from the others based on the URL of the requested resources. Another possible solution is to provide a user interface for the end users to enter information about their identity providers. With this approach, the application can start an SP-initiated SSO request with information about the IdP.

In the standard scenario, when a user requests access to a protected resource on the SP, common-domain browser cookies are used to determine where a user has authenticated in the past. Using this information, a PingFederate server can determine which IdP connection to use for sending an authentication request.

As an IdP Discovery provider, PingFederate can serve in up to three different roles: common domain server, common domain cookie writer, and common domain cookie reader. Each of these roles is necessary to support IdP Discovery. The roles can be distributed across multiple servers at different sites.

Common domain server

In this role the PingFederate server hosts a domain that its federation partners share in common. The common domain server allows partners to manipulate browser cookies that exist within that common domain. PingFederate can serve in this role exclusively or as part of either an IdP or an SP federation role, or both.

Common domain cookie writer

When PingFederate is acting in an IdP role and authenticates a user, it can write an entry in the common domain cookie, including its federation entity ID. An SP can look up this information on the common domain, not the same location as the common domain server described above.

Common domain cookie reader

When PingFederate is acting as an SP and needs to determine the IdPs with whom the user has authenticated in the past, it reads the common domain cookie. Based on the information contained in the cookie, PingFederate can then initiate an SSO authentication request using the correct IdP connection.

Steps

1. Go to the **Protocol Settings** window.
2. On the **IdP Discovery** tab, click **Configure IdP Discovery**.
3. On the **Domain Cookie Settings** tab, choose the discovery role or roles of your PingFederate server.
4. On the **Common Domain Service** tab, configure as follows.

Field	Description
Base URL of the PingFederate Common Domain Service	Enter the base URL of the PingFederate common domain service. A common domain service is where PingFederate reads or writes authentication information contained in shared cookies, as determined by whether your site is an SP or IdP, respectively. The service is shared if your PingFederate server is acting in both roles. You must use HTTPS for the common domain.
Pass Phrase and **Confirm Pass**	Enter and confirm the pass phrase that web applications must use to access the domain.

5. On the **Local Common Domain Server** tab, configure the required settings.

   A local common domain server is where PingFederate reads (as an SP) or writes (as an IdP) a common domain cookie (CDC) for IdP Discovery.

Field	Description
Common Domain	Enter the common domain. Your entry must include an initial period (.), as in the following example. `.example.com`
Cookie Lifetime (Days)	Enter the lifetime of the CDC in days. The range is `1` to `1825` days. To indicate a non-persistent session cookie, enter `-1`.
Pass Phrase and **Confirm Pass**	Enter and confirm the pass phrase that web applications must use to access the domain.

6. On the **Summary** tab, review and modify settings as needed. Then click **Save**.

   Result:

   The administrative console brings you back to the **IdP Discovery** tab.

7. Click **Next** and continue with the rest of the configuration.

   > **Tip**
   >
   > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

8. Perform one of the following actions to enable the setting of the common domain cookie at runtime:

   Choose from:

   - Make sure that, prior to launching any SSO events, the web application that implements IdP Discovery sets the cookie using the `/idp/writecdc.ping` application endpoint intended for that purpose.
   - Enable setting the cookie at runtime during SSO events by selecting the **IdP Discovery** check box on the **Connection Options** tab for the desired SP connection.

Reviewing protocol settings

You can review your protocol settings before saving them.

Steps

1. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
    - To keep your changes, click **Save**.
    - To discard your changes, click **Cancel**.

Administrative accounts

PingFederate supports five authentication schemes for administrative accounts.

The authentication schemes are:

- Native authentication
- LDAP authentication
- RADIUS authentication
- Certificate-based authentication
- OIDC-based authentication

For role-based access control, PingFederate provides two account types and four administrative roles, as shown in the following table.

PingFederate User Access Control

Account type	Administrative role	Access privileges
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Admin	Admin	Configure partner connections and most system settings, except the management of local accounts and the handling of local keys and certificates.
Admin	Expression Admin	Map user attributes by using the expression language, Object-Graph Navigation Language (OGNL).
Admin	Crypto Admin	Manage local keys and certificates.
Auditor	Not applicable	View-only permissions for all administrative functions. When the **Auditor** role is assigned, no other administrative roles can be set.

For native authentication, access and authorization are controlled by the local accounts defined on the **Administrative Accounts** window.

As needed, you can switch from native authentication to an alternative console authentication. Access and authorization are defined in the respective configuration file.

An administrative user can sign on from more than one browser or location. Moreover, multiple administrative users can sign on to the PingFederate administrative console at a time. You can optionally restrict the administrative console to one administrative user at a time by modifying the `pf.console.login.mode` property in the `<pf_install>/pingfederate/bin/run.properties` file. Regardless of the property configuration, any number of auditors can sign on at any time.

> **Note**
>
> For security, after three failed sign-on attempts from the same location within a short time period, the administrative console and the administrative API will temporarily lock out further attempts by the same user. The user must wait one minute to try again.

Local accounts defined on the **Administrative Accounts** window are shared between the administrative console and the administrative API if they are both configured to use native authentication, the default. If the administrative console is configured to use an alternative console authentication, the **Administrative Accounts** window appears only if the administrative API is left to use native authentication, and vice versa.

> **Tip**
>
> If you have connected PingFederate to PingOne for Enterprise, you can also single sign-on from the PingOne admin portal to the administrative console.

Related links

- Alternative console authentication

Enabling native authentication for the administrative console

When the administrative console is protected by native authentication, access is restricted to the local accounts you have defined.

Steps

1. Go to **System > Server** to open the **Administrative Accounts** window.
2. In the `<pf_insall>/pingfederate/bin/run.properties` file, change the value of the `pf.console.authentication` property to `pf.console.authentication=native`
3. Start or restart PingFederate.

Managing local accounts and role assignments

You can create, modify, update, or deactivate accounts in the **Administrative Accounts** window.

Steps

1. Go to **System > Server > Administrative Accounts**, and then perform any of the following actions.

Task	Steps
Create a local account	1. On the **Administrative Accounts** window, click **Create User**. 2. On the **User Information** tab, enter a username and other optional information. ⓘ **Note** If you want PingFederate to notify the user about password changes via email, you must supply an email address. 3. On the **Password Generation** tab, enter a password or click **Generate one-time password** to generate a random password for the account. ⓘ **Note** Upon successful authentication, the user will be required to change the password of the account immediately. 4. On the **Summary** tab, review your configuration, modify as needed, and then click **Done**. 5. On the **Administrative Accounts** window, select the applicable account type, **Auditor** or **Admin**, and one or more administrative roles for an **Admin** account. 6. Repeat these steps to create additional accounts.
Modify user information	1. On the **Administrative Accounts** window, select the account by its username. ⓘ **Note** Applicable only to active accounts. 2. On the **User Information** window, update the record, and then click **Done**. 3. Repeat these steps to update other accounts.
Update role assignments	1. Select a different account type, **Auditor** or **Admin**, for one or more accounts. 2. Select or clear the check boxes that correspond to the three administrative roles, **User Admin**, **Admin**, and **Crypto Admin** for one or more accounts. ⓘ **Note** Applicable only to the **Admin** accounts.

Task	Steps
Deactivate or reactive a native	1. Click **Deactivate** or **Activate** under **Action**. 2. Repeat this step to deactivate or reactive other accounts. ⓘ **Note** For traceability and accountability purposes, local accounts cannot be deleted Their records are retained and they can be reactivated if needed.

2. To keep your configuration, click **Save**.

Enabling notification messages for account management events

Administrators can optionally enable notifications for account management events. If enabled, PingFederate generates notification messages based on the following events.

About this task

See the following table for a description of an event and the corresponding alert.

Event	Alert
An administrator turns off the **Notify Administrator of Account Changes** option.	PingFederate generates a notification message to all administrators. The message includes the username of the administrator who made the change.
An administrator's email address is updated by another administrator.	PingFederate generates a notification message to the previous email address and another notification to the new email address. The message includes the username of the administrator who made the change.
An administrator's password is changed.	PingFederate generates a notification to the administrator whose password has been changed. The message includes the username of the administrator who made the change.

> ⓘ **Note**
>
> Account management events are only applicable when native authentication is enabled for the administrative console, the administrative API, or both in the `<pf_install>/pingfederate/bin/run.properties` file. If you are using an alternative console authentication, notifications, if any, such as password changes, are handled by the third-party system.

Steps

1. Go to **System > Server** to open the **Administrative Accounts** window.

2. Select the **Notify Administrator of Account Change** check box.

An email address must be provided for the applicable accounts.

3. Select a notification publisher instance from the list.

   If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

4. To keep your configuration, click **Save**.

Related links

- Managing notification publisher instances

Setting or resetting passwords

On the **Password Generation** tab, you can generate temporary passwords as you create local accounts for new users.

About this task

You can also assign temporary passwords for existing users who forget their passwords. Upon successful authentication, the users are required to change their passwords immediately.

Steps

1. Go to **System** > **Server** > **Administrative Accounts**.

2. To generate a random password for the account, on the **Password Generation** tab, enter a password or click **Generate one-time password**, and then click **Next**.

Changing passwords

Administrative users and auditors can change the passwords of their local accounts at any time.

About this task

> **Note**
>
> If you sign on to PingFederate using your network ID and password, you can change your password only at the network level. The new password will apply to PingFederate automatically the next time you log on.

Steps

1. Go to **System** > **Server** to open the **Administrative Accounts** window.

2. Click **Change Password** under **Action** for your account.

3. Enter your current password and new password twice in the related fields.

   > **Important**
   >
   > If you are the sole user administrator, take steps to ensure that you do not forget your new password.

4. To keep your configuration, click **Save**.

Configuring OIDC SSO to PingFederate from an external IdP

You can configure OpenID Connect (OIDC) single sign-on (SSO) for signing on to the PingFederate administrative console.

Before you begin

Make sure you have the following in place:

- A valid signing certificate. See Manage digital signing certificates and decryption keys.
- An openID and a profile scope. See Defining scopes.
- A policy contract with at least the following attributes: `sub`, `admin_role`, `iss`, `memberOf`. See Managing policy contracts.
- An identity provider (IdP) connection in your PingFederate instance with the following attributes: `SAML_SUBJECT`, `memberOf` – fulfilled by the policy contract authentication source. See Managing IdP connections.
- An service provider (SP) connection in your external IdP with the following attributes: `SAML_SUBJECT`, `memberOf` – fulfilled by whichever authentication source is appropriate and using whatever authentication flows you require (for example, username/password and multi-factor authentication (MFA)). See Accessing SP connections.

About this task

Configuring OIDC SSO for the PingFederate administrative console allows you to use an external IdP to authenticate administrative users. You can also use OIDC SSO to enable MFA because the administrative users are taken through an authentication policy flow that invokes an MFA adapter. Other console authentication types don't use authentication policies.

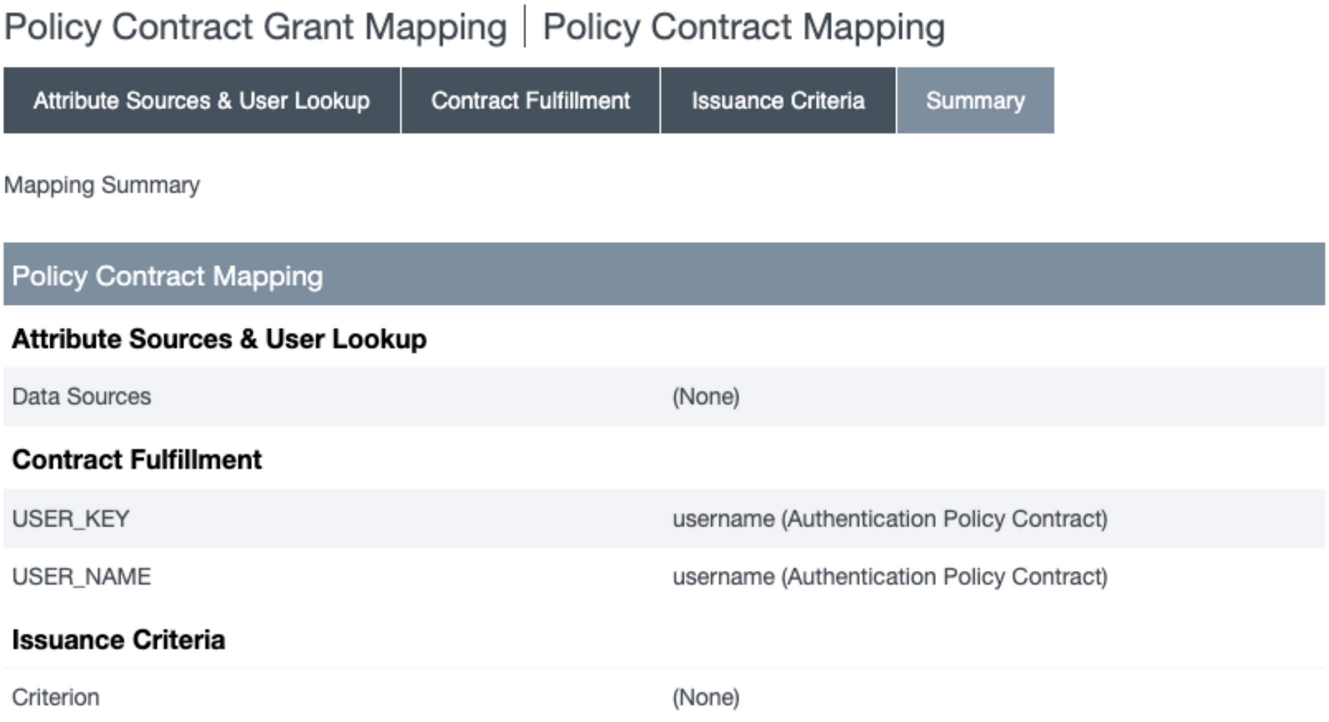
Mapping the policy contract for grant fulfillment

Complete the contract fulfillment.

Steps

1. Go to **Authentication** > **OAuth** > **Policy Contract Grant Mapping**.
2. In the **Policy Contract** menu, select your policy contract, and click **Add Mapping**.
3. On the **Attribute Sources & User Lookup** tab, click **Next**.
4. On the **Contract Fulfillment** tab, map the **User_Key** to the username from the policy contract.
    1. Beside **User_Key**, select **Authentication Policy Contract** in the **Source** menu and the policy contract's username in the **Value** menu.
    2. Beside **User_Name**, select **Authentication Policy Contract** in the **Source** menu and the policy contract's username in the **Value** menu.
    3. Click **Next**.
5. On the **Issuance Criteria** tab, click **Next**.

6. On the **Summary** tab, verify your configuration, and click **Save**.


Policy Contract Grant Mapping | Policy Contract Mapping
Attribute Sources & User Lookup
Contract Fulfillment
Issuance Criteria
Summary

Mapping Summary

Policy Contract Mapping

Attribute Sources & User Lookup

Data Sources	(None)

Contract Fulfillment

USER_KEY	username (Authentication Policy Contract)
USER_NAME	username (Authentication Policy Contract)

Issuance Criteria

Criterion	(None)

Configuring an access token manager

Create a JSON Web Token (JWT) access token management instance.

Steps

1. Go to **Applications** > **OAuth** > **Access Token Management**.

2. To create a new access token management instance, click **Create New Instance**.

3. On the **Type** tab:

    1. Enter a name for the instance in the **Instance Name** field and an ID in the **Instance ID** field.

    2. In the **Type** menu, select **JSON Web Tokens**.

    3. Click **Next**.

4. On the **Instance Configuration** tab:

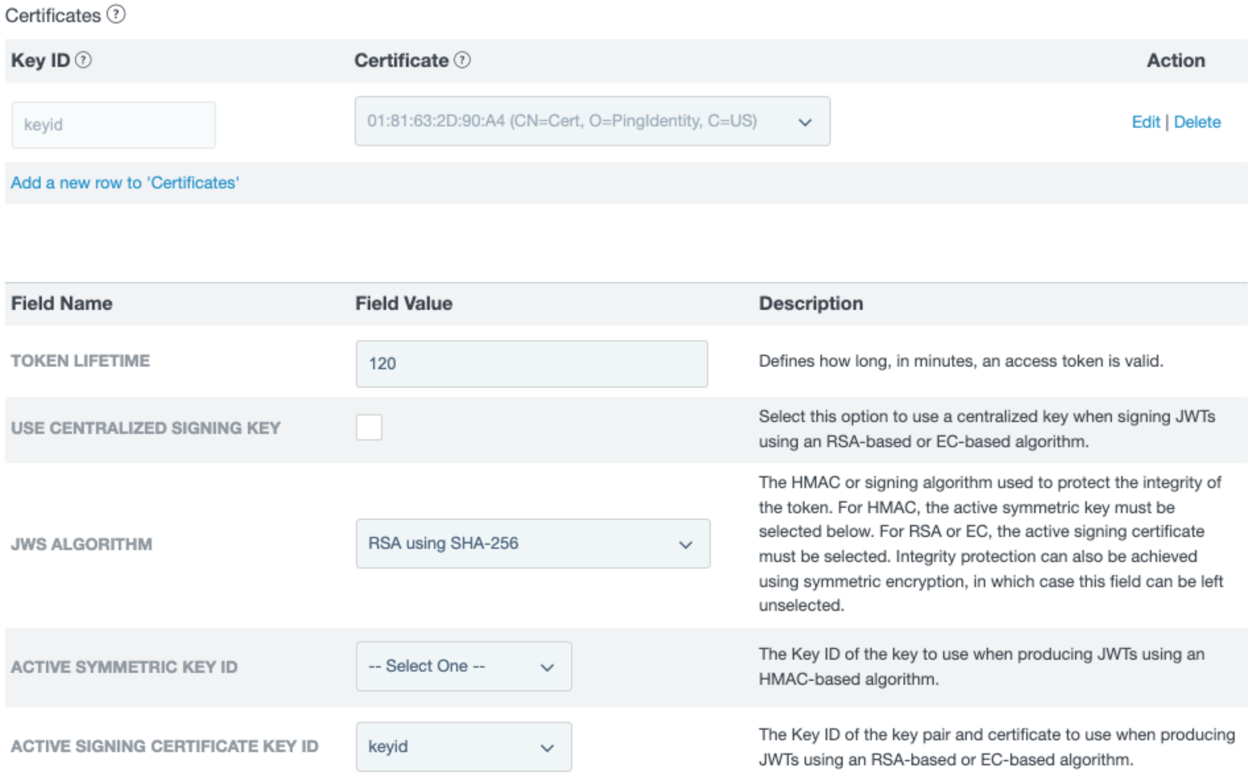
    1. In the **Certificates** section, click **Add a new row to 'certificates'**.

    2. In the **Key ID** field, enter an ID for the key.

    3. In the **Certificate** menu, select your signing certificate, and click **Update**.

    4. In the **JWS Algorithm** menu, select **RSA using SHA-256**.

5. In the **Active Signing Certificate Key ID** menu, select the key ID you entered in step b.



Certificates

Key ID	Certificate	Action
keyid	01:81:63:2D:90:A4 (CN=Cert, O=PingIdentity, C=US)	Edit | Delete

Add a new row to 'Certificates'

Field Name	Field Value	Description
TOKEN LIFETIME	120	Defines how long, in minutes, an access token is valid.
USE CENTRALIZED SIGNING KEY	□	Select this option to use a centralized key when signing JWTs using an RSA-based or EC-based algorithm.
JWS ALGORITHM	RSA using SHA-256	The HMAC or signing algorithm used to protect the integrity of the token. For HMAC, the active symmetric key must be selected below. For RSA or EC, the active signing certificate must be selected. Integrity protection can also be achieved using symmetric encryption, in which case this field can be left unselected.
ACTIVE SYMMETRIC KEY ID	-- Select One --	The Key ID of the key to use when producing JWTs using an HMAC-based algorithm.
ACTIVE SIGNING CERTIFICATE KEY ID	keyid	The Key ID of the key pair and certificate to use when producing JWTs using an RSA-based or EC-based algorithm.

6. Click **Next**.

5. On the **Session Validation** tab, click **Next**.

6. On the **Access Token Attribute Contract** tab:

   1. Make sure **User_Key** is selected in the **Subject Attribute Name** menu.

   2. In the **Extend the Contract** field, enter `admin_role`, and click **Add**.

   3. Repeat step b to add the `iss`, `memberOf`, and `sub` attributes.

   4. Click **Next**.

7. On the **Resource URIs** and **Access Control** tabs, click **Next**.

8. On the **Summary** tab, review your configuration. Click **Save**.

Configuring access token mapping

Map your policy contract context to the JWT access token manager.

Steps

1. Go go **Applications > OAuth > Access Token Mappings**.

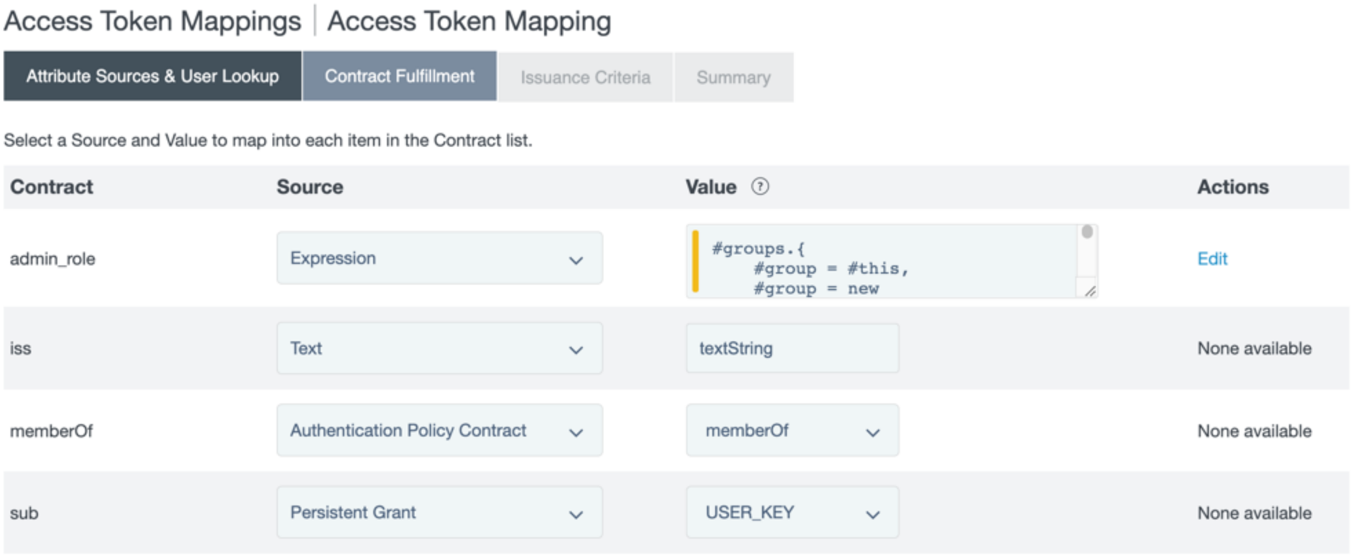
2. On the **Access Token Mappings** page in the **Context** menu, select your policy contract.

3. In the **Access Token Manager** menu, select your JWT ATM.

4. Click **Add Mapping**.

5. On the **Attribute Sources & User Lookup** tab, click **Next**.

6. On the **Contract Fulfillment** tab, select a **Source** and a **Value** to map into the `admin_role`, `iss`, `memberOf`, and `sub` attributes in the **Contract** list.


Access Token Mappings | Access Token Mapping
Attribute Sources & User Lookup
Contract Fulfillment
Issuance Criteria
Summary

Select a Source and Value to map into each item in the Contract list.

Contract	Source	Value	Actions
admin_role	Expression	#groups.{ #group = #this, #group = new	Edit
iss	Text	textString	None available
memberOf	Authentication Policy Contract	memberOf	None available
sub	Persistent Grant	USER_KEY	None available

   1. For the `admin_role` attribute, select **Expression** in the **Source** menu and, in the **Value** field, enter the following expression:

```
#filter1 = "^pf_admins.",
#filter2 = "^pf_cryptoadmins.",
#filter3 = "^pf_useradmins.*",
#role1 = "admin",
#role2 = "cryptoadmin",
#role3 = "useradmin",
#role4 = "expressionadmin",

#outboundattribute = new java.util.ArrayList(),

#groups = #this.get("apc.memberOf")!=null?#this.get("apc.memberOf").getValues():{},

#i = 0,

#groups.{
#group = #this,
#group = new javax.naming.ldap.LdapName(#groups[#i]),
#cn = #group.getRdn(#group.size() - 1).getValue().toString(),

#cn.matches(#filter1)?#outboundattribute.add(#role1):null,
#cn.matches(#filter1)?#outboundattribute.add(#role4):null,
#cn.matches(#filter2)?#outboundattribute.add(#role2):null,
#cn.matches(#filter3)?#outboundattribute.add(#role3):null,

#i = #i + 1},

#outboundattribute.size() > 0 ? new
org.sourceid.saml20.adapter.attribute.AttributeValue(#outboundattribute):null
```

> **Note**
>
> This example OGNL expression gets the `memberOf` value from the policy contract, looks for group distinguished name (DN) that match the filters, and assigns a role when a filter is matched. In the expression, anyone that is in the Admins group is assigned both the Admin and Expression Admin role, because the Expression Admin role requires the Admin role assignment. Using this expression to map roles allows you to control access with groups from your identity provider's data source. Match your filter values in the expression to the group names created in your LDAP directory to assign those roles.

2. For the `iss` attribute, select **Text** in the **Source** menu, and enter a text string in the **Value** field.

> **Note**
>
> Make a note of the text string. The value entered here is the issuer claim value and should identify the organization as the issuer.

3. For the `memberOf` attribute, select **Authentication Policy Contract** in the **Source** menu, and **memberOf** in the **Value** menu.

4. For the `sub` attribute, select **Persistent Grant** in the **Source** menu, and **USER_KEY** in the **Value** menu.

5. Click **Next**.

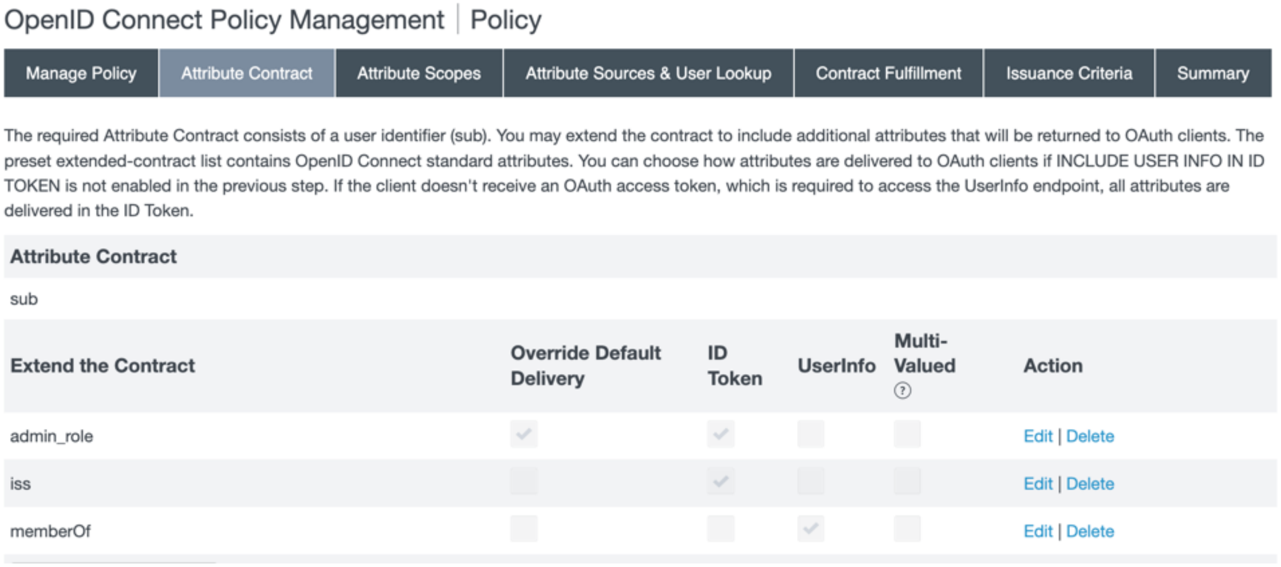
7. On the **Issuance Criteria** tab, click **Next**.

8. On the **Summary** tab, review your mappings. Click **Save**.

Creating the OIDC policy

Steps

1. Go to **Applications** > **OAuth** > **OpenID Connect Policy Management**.
2. Click **Add Policy**.
3. On the **Manage Policy** tab:
    1. In the **Policy ID** field, enter the policy identifier.
    2. In the **Name** field, enter the policy name.
    3. In the **Access Token Manager** menu, select your JWT access token manager.
    4. Click **Next**.
4. On the **Attribute Contract** tab, add the `admin_role`, `iss`, and `memberOf` attribute contracts.
    1. In the **Extend the Contract** field, enter `admin_role`, and click **Add**.
    2. Repeat step a. to add the `iss` and `memberOf` attributes.
    3. Click the **Edit** action for `admin_role`. Select the **Override Default Delivery** and **ID Token** check boxes, then click the **Update** action.
    4. Repeat step c for `iss`, selecting the **ID Token** check box, and for `memberOf`, selecting the **UserInfo** check box.



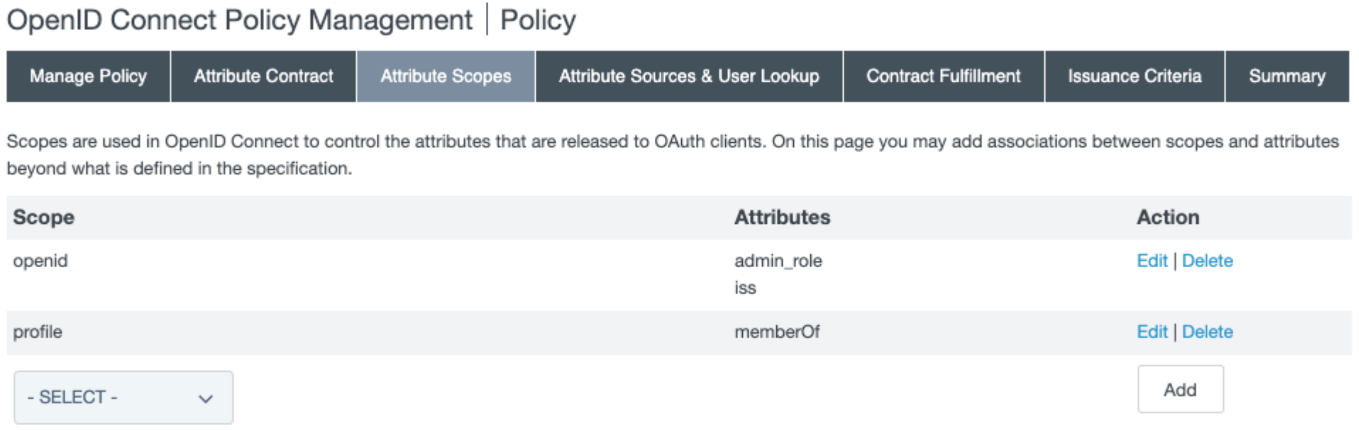
The required Attribute Contract consists of a user identifier (sub). You may extend the contract to include additional attributes that will be returned to OAuth clients. The preset extended-contract list contains OpenID Connect standard attributes. You can choose how attributes are delivered to OAuth clients if INCLUDE USER INFO IN ID TOKEN is not enabled in the previous step. If the client doesn't receive an OAuth access token, which is required to access the UserInfo endpoint, all attributes are delivered in the ID Token.

Attribute Contract
sub

Extend the Contract	Override Default Delivery	ID Token	UserInfo	Multi-Valued	Action
admin_role	☑	☑	□	□	Edit | Delete
iss	□	☑	□	□	Edit | Delete
memberOf	□	□	☑	□	Edit | Delete

    5. Click **Next**.

5. On the **Attribute Scopes** tab, add the `admin_role` and `iss` attributes to the openid scope and the `memberOf` attribute to the profile scope.



OpenID Connect Policy Management | Policy

Scopes are used in OpenID Connect to control the attributes that are released to OAuth clients. On this page you may add associations between scopes and attributes beyond what is defined in the specification.

Scope	Attributes	Action
openid	admin_role iss	Edit | Delete
profile	memberOf	Edit | Delete
- SELECT -		Add

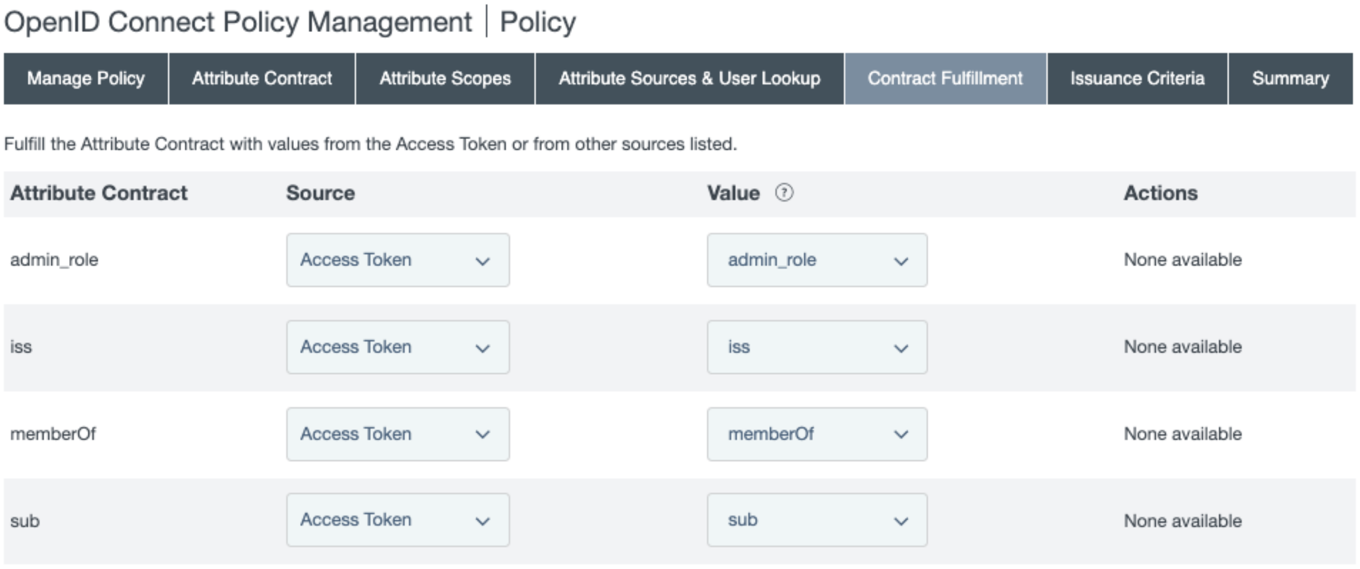
   1. In the **Scope** menu, select **openid**. Select the `admin_role` attribute's check box, and click **Add**. The `iss` attribute should already be selected.

   2. In the **Scope** menu, select **profile**. Select the `memberOf` attribute's check box, and click **Add**.

   3. Click **Next**.

6. On the **Attribute Sources & User Lookup** tab, click **Next**.

7. On the **Contract Fulfillment** tab, select a **Source** and a **Value** to map into the `admin_role`, `iss`, `memberOf`, and `sub` items in the **Attribute Contract** list.



OpenID Connect Policy Management | Policy

Fulfill the Attribute Contract with values from the Access Token or from other sources listed.

Attribute Contract	Source	Value	Actions
admin_role	Access Token	admin_role	None available
iss	Access Token	iss	None available
memberOf	Access Token	memberOf	None available
sub	Access Token	sub	None available

   1. For the `admin_role` attribute contract, select **Access Token** in the **Source** menu and **admin_role** in the **Value** menu.

   2. For the `iss` attribute contract, select **Access Token** in the **Source** menu and **iss** in the **Value** menu.

   3. For the `memberOf` attribute contract, select **Access Token** in the **Source** menu and **memberOf** in the **Value** menu.

   4. For the `sub` attribute contract, select **Access Token** in the **Source** menu and **sub** in the **Value** menu.

   5. Click **Next**.

8. On the **Issuance Criteria** tab, click **Next**.

9. On the **Summary** tab, review your configuration. Click **Save**.

Creating the client

Steps

1. Go to **Applications** > **OAuth** > **Clients**, and click **Add Client**.

2. In the **Client ID** field, enter an ID for the client.

3. In the **Name** field, enter a name for the client.

4. In the **Client Authentication** section, select **Client Secret**.

5. In the **Client Secret** section, select the **Change Secret** check box and enter a password in the field. If preferred, you can click **Generate Secret**.

6. In the **Redirect URIS** field, enter the following URI, and click **Add**. `https://<your PingFederate admin server hostname>/pingfederate/app?service=finishsso`

7. For **Bypass Authorization Approval**, select the **Bypass** check box.

8. For **Allowed Grant Types**, select the **Authorization Code** check box.

9. In the **Default Access Token Manager** menu, select your JSON access token manager.

10. For **Restrict to Default Access Token Manager**, select the **Restrict** check box.

11. In the **OpenID Connect** section, in the **Policy** menu, select your OIDC policy.

12. Click **Save**.

Creating an OAuth Set Authentication selector

Create an OAuth Set Authentication selector and add your client to it.

Steps

1. Go to **Authentication** > **Policies** > **Selectors**.

2. On the **Selectors** page, click **Create New Instance**.

3. On the **Type** tab, do the following.

    1. In the **Instance Name** field, enter a name for the instance.

    2. In the **Instance ID** field, enter an ID for the instance.

    3. In the **Type** menu, select **OAuth Client Set Authentication Selector**.
    4. Click **Next**.
4. On the **Authentication Selector** tab:
    1. Click **Add a new row to Clients**.
    2. In the **Client ID** menu, select your client, and click the **Update** action.
    3. Click **Next**.
5. On the **Summary** tab, review your configuration. Click **Save**.

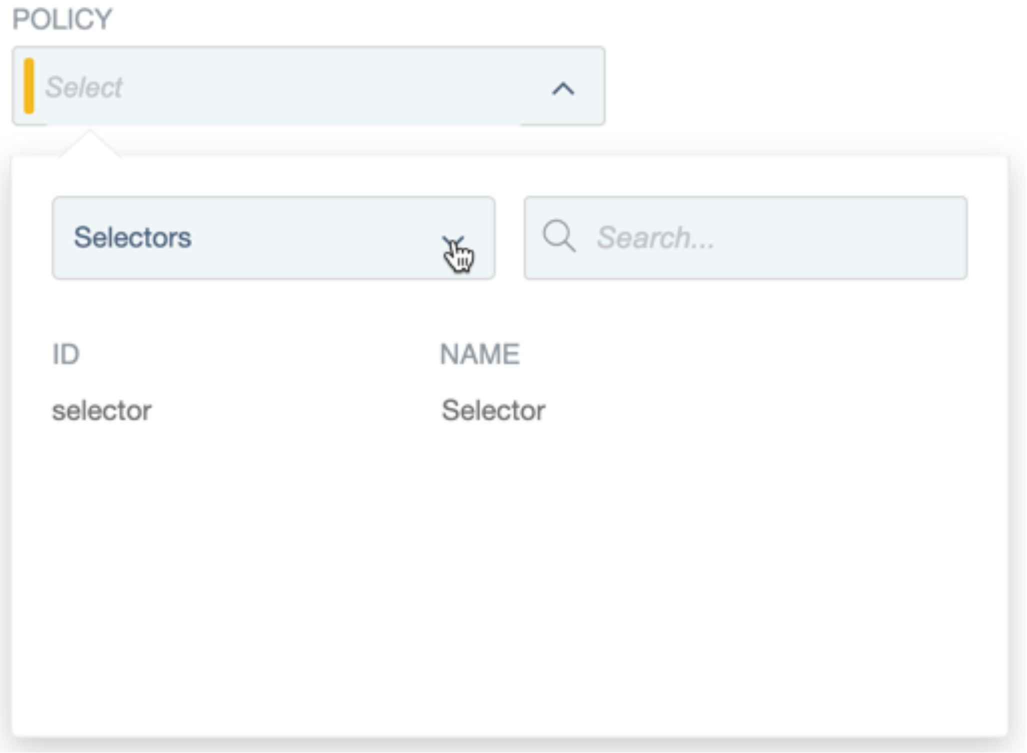
Creating an authentication policy

Create an authentication policy that is triggered by the selector, sends the user to the external IdP, and fulfills the policy contract.

Steps

1. Go to **Authentication > Policies > Policies**.
2. Click **Create New Instance**.
3. Click **Add Policy**.
4. On the **Policy** page In the **Name** field, enter a name for the policy.
5. **Optional:** In the **Description** field, enter a description for the policy.

6. Click in the **Policy** field, and select **Selectors** in the menu.



7. Select your selector.

8. For the **No** option, click **Continue**.

9. For the **Yes** option, select **IdP Connections** in the menu, and select your IdP connection.

10. For the **Fail** option, click **Done**.

11. For the **Success** option, select **Policy Contracts** in the menu, and select your policy contract.

12. Click **Contract Mapping**.

13. On the **Attribute Sources & User Lookup** tab, click **Next**.

14. On the **Contract Fulfillment** tab, map the `memberOf`, `subject`, and `username` attributes. If your policy contract has additional attributes, select **No Mapping** in the **Source** menu for those attributes.

    1. For the `memberOf` attribute, select **IdP Connection** in the **Source** menu and `memberOf` in the **Value** menu.

    2. For the `subject` attribute, select **IdP Connection** in the **Source** menu and **SAML_SUBJECT** in the **Value** menu.

    3. For the `username` attribute, select **IdP Connection** in the **Source** menu and **SAML_SUBJECT** in the **Value** menu.

15. On the **Issuance Criteria** tab, click **Next**.

16. On the **Summary** tab, review your configuration. Click **Done**.

17. On the **Policy** page, click **Done**.

18. On the **Policies** tab, click **Save**.

Exporting the signing certificate

When your PingFederate server is redirected to its own token endpoint, it must trust the certificate used to sign its tokens.

Steps

1. Go go **Security** > **Certificate & Key Management** > **Signing & Decryption Keys & Certificates**.
2. In the **Action** menu for the certificate that PingFederate will use to sign its tokens, select **Export**.
3. On the **Export Certificate** tab, select **Certificate Only**, and click **Next**.
4. On the **Export & Summary** tab, click **Export**.
5. Click **Done**.

Importing the certificate to trusted CAs

After you have exported the signing certificate, you must import it to trusted certificate authorities.

Steps

1. Go to **Security** > **Certificate & Key Management** > **Trusted CAs**.
2. Click **Import**.
3. On the **Import Certificate** tab, click **Choose File**.
4. Navigate to the location of the certificate that you exported, select it, and click **Open**.
5. On the **Import Certificate** page, click **Next**.
6. On the **Summary** tab, review your certificate information, and click **Save**.

Configuring properties files

Configure required parameters in PingFederate’s `oidc.properties` and `run.properties` files.

Steps

1. Configure the required parameters in the `<pf_install>/pingfederate/bin/oidc.properties` file.

   You’ll need the client ID and secret from the client you created, and you should obfuscate the secret. You’ll also need the `iss` attribute value you used in the access token manager mappings.

   Use the authorization and token endpoints with your PingFederate base URL.

   Example:

An example configuration is shown here:

```
client.id=pfadminconsole
client.authn.method=client_secret_basic
client.secret=***
authorization.endpoint=https\://pingfed-idp.ad.jibboo.org\:9031/as/authorization.oauth2
token.endpoint=https\://pingfed-idp.ad.jibboo.org\:9031/as/token.oauth2
issuer=jibbooidp
scopes=openid
username.attribute.name=sub
role.attribute.name=admin_role
role.admin=admin
role.cryptoManager=cryptoadmin
role.userAdmin=useradmin
role.expressionAdmin=expressionadmin
```

2. Configure the `pf.console.authentication` parameter in the `<pf_install>/pingfederate/bin/run.properties` file as follows: `pf.console.authentication=oidc`

3. Restart your PingFederate server.

License management

PingFederate licensing is handled differently depending on whether you are installing and setting up PingFederate for the first time, or upgrading your existing PingFederate installation to a later version.

Initial PingFederate installations

During the initial setup for a new PingFederate installation, you are prompted to upload a license file.

Depending on your licensing agreement, your license might have an expiration date. If your license key is going to expire, or has expired recently, you can import a new license file to replace the existing license key through the administrative console.

The administrative console displays a warning message ahead of the expiry of your license. Optionally, you can configure PingFederate to notify the administrators ahead of the license expiration date.

PingFederate upgrades

If you choose to upgrade PingFederate using the Upgrade Utility, your current PingFederate license is automatically copied to the target installation if it is valid. If your current license is not valid, you must obtain a new license and specify its full path and filename when performing the upgrade.

If you choose not to use the Upgrade Utility, you must specify the full path and filename of a valid license when performing the upgrade.

Reviewing license information

You can review a summary of your PingFederate license in the License window.

About this task

> **Important**
>
> If the license specifies an expiration date, the license expires at the beginning of that day.

Steps

1. On the Help menu, click About.
2. In the dialog box, click View / Update License. The License window opens.

   > **Tip**
   >
   > Another way to open the **License** window is to go to **System** > **Server** > **License**.

Generate a new license key

You can generate a new license key in the Ping Identity support portal.

Steps

1. Sign on to the Ping Identity support portal.
2. Click Manage License Keys.
3. On an existing license key, click View.
4. Click Request New Key.
5. On the confirmation message, click Confirm.

Result

The new license key displays in your key list.

Installing a license key on a new or upgraded PingFederate server

You can install a license key on the PingFederate server from the administrative console.

Steps

1. Start PingFederate and sign on to the administrative console.
2. Go to System > Server to open the License Managementwindow, and then click Choose file to select the license file. Click Import.

   Result:

   If the license is for the wrong version of PingFederate or is found to be invalid for some other reason, PingFederate displays an error message.

   When the license file is verified for use with your PingFederate instance, the license information is saved in the `<pf_install>/pingfederate/server/default/conf/pingfederate.lic` file.

3. If you have a clustered PingFederate environment, go to System > Server to open the Cluster Management window, and then click Replicate Configuration.

   > **Note**
   >
   > You must use the **Replicate Configuration** window to initiate the transmission of the license file from the console node to all engine nodes. As an added measure, the administrative console reminds you to do so as well.

   When an engine node receives the license information from the console node, it saves the new license information to the `<pf_install>/pingfederate/server/default/conf/pingfederate.lic` license file.

   For any engine node that was offline at the time of the import, when it restarts and joins the cluster, it consumes the new license information from the console node and applies the same processing logic.

   > **Note**
   >
   > PingFederate no longer maintains its license information in the `<pf_install>/pingfederate/server/default/data/.pingfederate.lic` file, which is referred to as the secondary license file in the previous versions of PingFederate. The `.pingfederate.lic` file, if any, is ignored.

Installing a replacement license key

You can replace your existing PingFederate license key by importing a new license file using the PingFederate administrated console.

Steps

1. Start PingFederate and sign on to the administrative console.

2. Go to System > Server > License.

3. In the License window, click Choose File, select the license file, and click Import.

   Result:

   If the license is for the wrong version of PingFederate or is found to be invalid for some other reason, PingFederate displays an error message and keeps the existing license regardless of whether the existing license has expired.

   If the new license does not include support for features found in your existing license, or if there is some other potential problem with the license, PingFederate advises and prompts you on whether to continue.

   When the license file is verified for use with your PingFederate instance, the license information is saved in the `<pf_install>/pingfederate/server/default/conf/pingfederate.lic` file.

   The previous license file is renamed with a timestamp in the same `conf` directory.

4. If you have a clustered PingFederate environment, go to System > Server > Cluster Management and click Replicate Configuration.

> **Note**
>
> You must use the **Replicate Configuration** window to initiate the transmission of the license file from the console node to all engine nodes.

When an engine node receives license information from the console node, if the issue date of the new license is not as recent as that of the existing license, the engine node ignores the new license from the console and logs the following warning message in the server log.

```
License from Console node ignored as Engine node has recently obtained license.
```

> **Tip**
>
> If you prefer to use the license from the console node, even the existing license on the engine node is more recent in terms of the issue date, manually remove or rename the `<pf_install>/pingfederate/server/default/conf/pingfederate.lic` license file on the engine node, then click **Replicate Configuration** on the **Cluster Management** window again.

If the issue date of the new license is more recent than or equal to that of the existing license, the engine node saves the new license information to the `<pf_install>/pingfederate/server/default/conf/pingfederate.lic` license file and activates it immediately. No restart is required.

For any engine node that was offline at the time of the import, when it restarts and joins the cluster, it consumes the new license information from the console node and applies the same processing logic.

> **Note**
>
> PingFederate no longer maintains its license information in the `<pf_install>/pingfederate/server/default/data/.pingfederate.lic` file, which is referred to as the secondary license file in the previous versions of PingFederate. The `.pingfederate.lic` file, if any, is ignored.

Configuring notification for licensing events

If your PingFederate license has an expiration date, you can configure PingFederate to notify the administrators ahead of the license expiration date to minimize potential service disruptions.

About this task
Steps

1. Go to System > Monitoring & Notifications to open the Runtime Notifications window.
2. Select the Notification for Servers Licensing Events check box.

   > **Note**
   >
   > This check box appears only if your PingFederate license has an expiration date.

3. Enter the email address of the intended recipient.
4. Select a notification publisher instance from the list.

If you have not yet configured the desired notification publisher instance, click Manage Notification Publishers.

5. To keep your configuration, click Save.

Related links

- Managing notification publisher instances

Configuration archive

You can use configuration archives as backup files for the current PingFederate installation.

> **Note**
>
> In addition to backup, you can use configuration archives for disaster recovery purposes.
>
> - If the console server is still functional, you can import a recent configuration archive to solve the problem.
> - In a clustered PingFederate
>
> Using a configuration archive is not necessary in a clustered environment where the console server is still functional and some of the engine nodes are gone. In this case, create new engine nodes and then replicate the configuration from the console node. environment, if the console server and the engine nodes are all gone, you can import a recent configuration archive to a new console server and then replicate the configuration to new engine nodes. All other configurations that occurred outside of the archive will have to be redone manually.

If changes were made to the configuration since the last backup archive was created, the next time an administrator signs on to the administrative console or imports an existing archive, PingFederate automatically creates a time-stamped configuration (`.zip`) archive. The archives are stored in the `<pf_install>/pingfederate/server/default/data/archive` directory.

The automatic backup process typically completes without delays. For deployments with hundreds of connections or OAuth clients, or both, administrators can configure PingFederate to create configuration archives periodically instead.

Additionally, administrators can export the current configuration to a `.zip` file in the Configuration Archive window. This window is only available to administrators whose accounts have been assigned the User Admin, Admin, Crypto Admin, and Expression Admin roles.

> **Note**
>
> The Expression Admin role must be assigned to give administrators sufficient permissions to create configuration archives.

> **Warning**
>
> The backup file contains your complete PingFederate configuration. To protect your data, confirm the backup file is protected with appropriate security controls in place before exporting it.
> Sharing the archive is a security risk because the private keys are stored in the archive. An archive should only be shared if the security of that PingFederate instance is not important, such as a development or test environment.

On the Configuration Archive window, administrators can import an existing archive for immediate deployment into a running PingFederate server.

Administrators can also deploy a configuration archive manually by copying the `.zip` file to the `environment`, `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory. After copying the `.zip` file, it must be renamed to `data.zip`.

> **Note**
>
> If you use the drop-in deployment process:
>
> - PingFederate will not let you import the configuration archive of an older or newer version, and to ensure successful importation of the configuration archive file with this process, you must rename the file `data.zip`.
> - On startup, the heartbeat endpoint will not return `200` until the archive import completes. If you have configured a health check or probe that can trigger a restart of the server, crash loop behavior can result. Review the configuration of these checks to ensure time thresholds are set appropriately.

Configuration archives are intended for administrative-console configuration only. The following files are not included in the archives:

- Launch scripts in the `<pf_install>/pingfederate/bin` and `<pf_install>/pingfederate/sbin` directories.
- Web container configuration files in the `<pf_install>/pingfederate/etc` directory.
- Log files in the `<pf_install>/pingfederate/log` directory.
- Database drivers and program files from adapters and any other plugins in the `<pf_install>/pingfederate/server/default/lib` and `<pf_install>/pingfederate/server/default/deploy` directories.
- Other files, including the license file, the advanced cluster configuration files, and the user-facing email and HTML templates, in the `<pf_install>/pingfederate/server/default/conf` directory.

  If any changes have been made to files that are not part of the configuration archive, those files must be preserved manually.

  > **Tip**
  >
  > You can export a configuration archive, extract the `.zip` file, and determine whether specific files are part of the configuration archive, or not.

  > **Important**
  >
  > Draft connections in archives are not imported. Complete any unfinished partner connections if you want to include them in a full backup archive or in an archive to be used for configuration migration.

Configuring a backup schedule

For deployments with hundreds of connections or OAuth clients, or both, administrators can configure PingFederate to create configuration archives periodically.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.domain.mgmt.impl.DataArchiveBackup.xml` file. See the following table for each property.

Property	Description
`ScheduledBackupEnabled`	When set to `true`, PingFederate creates a configuration archive daily if configuration has changed since the creation of the last archive. This is true regardless of the backup method used. The default value is `false`.
`BackupTime`	The local time at which PingFederate creates a configuration archive when the `ScheduledBackupEnabled` property is set to `true`. Ignored when the `ScheduledBackupEnabled` is set to `false`. The default value is `00:00:00`, which represents midnight.
`NumTriesWithIntegrityFailure`	Upon the creation of the scheduled configuration archive, if PingFederate detects a configuration change during the scheduled backup process, it retries immediately. The `NumTriesWithIntegrityFailure` property indicates the maximum number of attempts. Ignored when the `ScheduledBackupEnabled` is set to `false`. The default value is `3`.
`BackupOnAdminLogin`	When set to `false`, PingFederate does not create a configuration archive when an administrator signs on. Regardless of the value of this property, PingFederate always creates a configuration archive before an archive is imported. The default value is `true`.
`MaxOldArchiveFiles`	The number of older configuration archives that PingFederate keeps. When the limit is reached, the oldest file is removed. The default value is `25` files.

The `ScheduledBackupEnabled` and `BackupOnAdminLogin` properties are not mutually exclusive. For example, if your deployment has 50 connections, you can enable both automatic and scheduled backup processes.

Note

If you have a clustered PingFederate environment, edit the `org.sourceid.saml20.domain.mgmt.impl.DataArchiveBackup.xml` file on the console node.

Exporting an archive

On the Configuration Archive window, you can export the current administrative-console configuration to a `.zip` file. This window is only available to administrators whose accounts have been assigned the User Admin, Admin, Expression Admin, and Crypto Admin roles.

Steps

- On the Export tab, click Export, then save the `.zip` file.

> ⚠ **Warning**
>
> The backup file contains your complete PingFederate configuration. To protect your data, confirm the backup file is protected with appropriate security controls in place before exporting it.
> Sharing the archive is a security risk because the private keys are stored in the archive. An archive should only be shared if the security of that PingFederate instance is not important, such as a development or test environment.

Importing and deploying administrative console configuration data

On the Configuration Archive window, you can import and deploy administrative-console configuration data from a `.zip` file.

About this task

When you import and deploy configuration data using the Import tab, PingFederate displays error messages if there are any missing plugin components, such as adapters, database drivers, or token translators, on which the archive depends, or any mismatches of PingFederate licensing authorization. You can choose to force the deployment and then install the missing components later.

> ⓘ **Note**
>
> Installation of any missing database drivers or other third-party libraries will require a restart of PingFederate.

> ⓘ **Caution**
>
> Deploying a configuration archive, either manually or by using the administrative console, always overwrites all existing configuration data.

To import and deploy administrative-console configuration data from an archive:

Steps

1. On the Configuration Archive window's Import tab, choose the desired configuration archive from your system.
2. Optional: If you want PingFederate to deploy the archive regardless of whether dependency errors are detected, select the Force Import check box.

> ◇ **Important**
>
> If you enable this feature, consult the server start-up console or the server log for any messages concerning missing plugin components or other errors.

3. Optional: To re-encrypt imported configuration data, select the Re-encrypt Data check box. PingFederate will re-encrypt sensitive imported information, such as datastore passwords and adapter shared secrets, with the primary configuration encryption key. This is most useful when you don't want to share configuration encryption keys between the two environments.

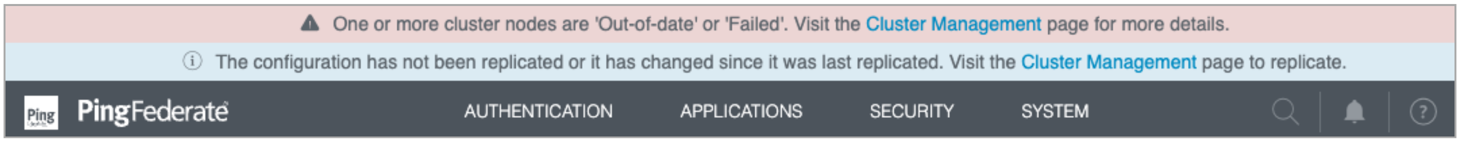
> **Important**
>
> If you re-encrypt an archive, you might lose access to external OAuth clients and runtime state, such as grants or sessions, that were encrypted with keys from the archive. PingFederate will not import configuration encryption keys in the data archive. Also, it will not re-encrypt data stored externally, such as OAuth clients, persistent grants, or persistent sessions in an external datastore.

4. Click Import.

Cluster management

When multiple PingFederate servers are running as a cluster, the administrative console provides a Cluster Management window that you can use to ensure the all the nodes have the same configuration and license.

When you change the configuration of PingFederate on the administrative console, a notification banner appears at the top of the console, reminding you to go to the Cluster Management window and replicate the configuration of the administrative console node to all engine nodes in the cluster. The replication procedure generates new replication data that the engine nodes use to replicate from the administrative node's configuration. Another notification banner appears when a cluster node is out-of-date or has failed.



One or more cluster nodes are 'Out-of-date' or 'Failed'. Visit the Cluster Management page for more details.

The configuration has not been replicated or it has changed since it was last replicated. Visit the Cluster Management page to replicate.

Each node periodically reaches out to a random node in the cluster to see if newer replication data is available. If a node finds newer data, it attempts to retrieve and apply it. By default, the nodes poll another node every 60 seconds. You can change the replication poll interval by editing the value of the `replication.poll.interval` property in the `<pf_install>/pingfederate/server/default/conf/cluster-config-replication.conf` file.

The Cluster Management window also shows the version of each node.

The following table describes the properties in the `<pf_install>/pingfederate/server/default/conf/cluster-config-replication.conf` file.

Property description

Property	Description
`rpc.replication.status.timeout`	The timeout period for replication status requests, in milliseconds. The default value is `2000`.
`rpc.replication.data.timeout`	The timeout period for requests to retrieve configuration data, in milliseconds. The default value is `20000`.

Property	Description
`replication.data.retries`	The number of retry attempts, if an error occurs while retrieving configuration data. The default value is `1`.
`replication.poll.interval`	The interval in seconds to check if more recent configuration data is available The default value is `60`.
`require.replication.data.on.startup`	Indicates if engines require replication data for startup and serving traffic. When set to true, engines that fail to retrieve replication data from the cluster during startup will exit, preventing them from serving traffic. NOTE: If an engine has previously retrieved replication data from a console or engine and is restarted, this flag will not be enforced. Instead, the engine will start up with the previously downloaded data. The default value is `false`.
`publish.replication.data.on.startup`	Enables availability of previously published replication data (in the case of consoles) or retrieved data (in the case of engines) to the cluster during startup, eliminating the need for manual or automated replication triggers upon console restart. **Note** This feature relies on the saved replication data being accessible after node restarts. If the `data` directory does not persist across restarts, a restarted node will have no replication data to publish. To benefit from this flag, the console's `data` directory must persist across restarts at a minimum. The default value is `true` for fresh installations and `false` for upgrades.

Related links

- Server Clustering Guide

Replicating configurations

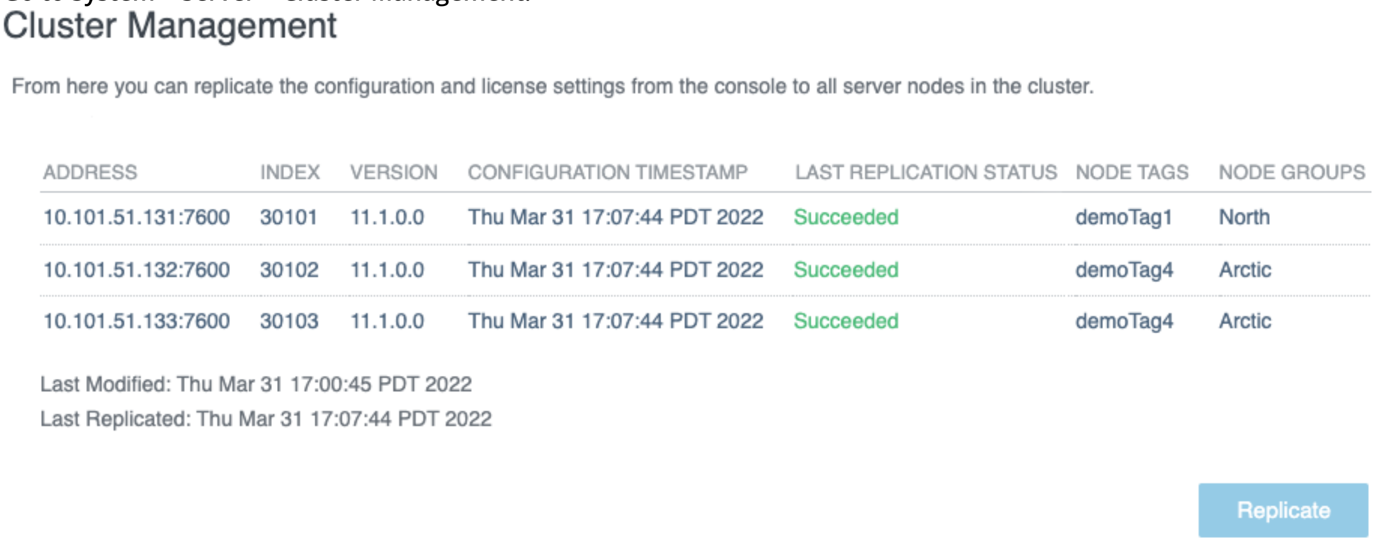
Use the Cluster Management window to replicate the configuration and license settings from the PingFederate administrative console node to the cluster's engine nodes.

About this task

The replication process takes only a moment. The engine nodes continue to process requests coming while they load updated replication data into memory. When the reload is complete, the updated configuration is activated for new requests.

Steps

1. Go to System > Server > Cluster Management.



Cluster Management

From here you can replicate the configuration and license settings from the console to all server nodes in the cluster.

ADDRESS	INDEX	VERSION	CONFIGURATION TIMESTAMP	LAST REPLICATION STATUS	NODE TAGS	NODE GROUPS
10.101.51.131:7600	30101	11.1.0.0	Thu Mar 31 17:07:44 PDT 2022	Succeeded	demoTag1	North
10.101.51.132:7600	30102	11.1.0.0	Thu Mar 31 17:07:44 PDT 2022	Succeeded	demoTag4	Arctic
10.101.51.133:7600	30103	11.1.0.0	Thu Mar 31 17:07:44 PDT 2022	Succeeded	demoTag4	Arctic

Last Modified: Thu Mar 31 17:00:45 PDT 2022
Last Replicated: Thu Mar 31 17:07:44 PDT 2022

2. Click Replicate.

   Result:

   PingFederate generates new replication data, updates the Last Replicated timestamp, and notifies the engine nodes that a new configuration is available. Then each node retrieves and applies the replication data, quickly going through the following Replication Status states:

   - Out-of-Date
   - Retrieving
   - Applying
   - Succeeded

Troubleshooting

If, after a few minutes, the Replication Status of any node is Failed, investigate if a network communication failure, improper file permissions at the node, or another problem is causing the replication failure.

Virtual host names

You can optionally define a list of alternate domain names at which PingFederate receives application and protocol messages.

This is done in the Virtual Host Names window. When configured, PingFederate honors the originally requested host throughout all browser redirects and metadata retrieval if the requested host matches one of the virtual host names. This capability allows you to fully support any number of branded URLs regardless of configured use cases within a single PingFederate environment.

Furthermore, virtual host names allow more flexibility for validating protocol elements, such as the `Destination` and `Recipient` elements in SAML inbound messages and the `aud` claim in JSON web tokens (JWTs) received from OAuth clients for client authentication purpose.

SAML inbound message

In certain contexts, the SAML specifications require that XML messages include a URL identifying the host name to which the sender directed the message. As the recipient of such messages, PingFederate validates that the value matches the location where the message is received, which is the Base URL value defined in the Protocol Settings window on the Federation Info tab.When virtual host names are configured, PingFederate takes them into consideration as part of its message-security validation process, in addition to its base URL.

OAuth client authentication using the private_key_jwt client authentication method

An OAuth client can authenticate with an authorization server by presenting a signed JWT. Per specification, the client must include the intended recipient as the `aud` claim value in its JWT. When acting as the authorization server, PingFederate verifies that the destination of the `aud` claim value matches either its base URL or the Token Endpoint Base URL value defined in the Authorization Server Settings window.When virtual host names are configured, PingFederate uses them in its verification process as well.

> **Note**
>
> Virtual host names and virtual server IDs serve different purposes. The latter provides separate unique identifiers on a per-connection basis for a federation deployment, normally in the same domain. For more information, see Multiple virtual server IDs. Virtual host names and virtual server IDs are not mutually exclusive. Depending on your use cases and infrastructure, you can configure both virtual server IDs and virtual host names in your PingFederate environment.

Multiple site certificates

When multiple domain names are involved, you can configure PingFederate with multiple site certificates so that PingFederate can serve a different site certificate based on the requested host. For more information, see Manage SSL server certificates.

Configuring virtual host names

Use the PingFederate administrative console to configure and manage virtual host names.

Steps

1. Go to System > Server to open the Virtual Host Names window.

2. Perform any of the following actions.

Option	Action
Add a new entry	Enter the desired value and click Add
Modify an existing entry	Click Edit or Update
Remove an existing entry	Click Delete
Keep your changes	Click Save

Option	Action
Discard your changes	Click Cancel

Extended properties

Add any number of extended properties to store additional information about connections, OAuth clients, or both.

You add these extended properties on the Extended Properties page as described in Defining extended properties. When adding an extended property, you can define it as a single-value property or a multivalued property. These extended properties become available to all connections and clients and, if defined, will be passed to all applicable velocity templates and as a request context parameter in the authentication API. As you create or update a connection or a client, you can populate values for any of them. For OAuth clients, if dynamic client registration is configured and enabled, developers can populate extended property values by including them in the client registrations.

Authentication policies

You can leverage extended properties to drive authentication experience and requirements by configuring an instance of the Extended Property Authentication Selector for each property that matters, placing this selector instance in an authentication policy, and defining a policy path for each selector result value. At runtime, PingFederate routes browser-based single sign-on (SSO) requests, OAuth authorization requests, and OAuth grant management requests to the desired authentication sources based on the applicable policy.

For more information, see Configuring the Extended Property Authentication Selector.

OAuth attributes fulfillment and issuance criteria

You can use extended properties as attribute sources when fulfilling persistent grants and token contracts. You can also define issuance criteria to verify extended property values.

Defining extended properties

You can define extended properties using the PingFederate administrative console.

Steps

1. Go to System > Server > [.wintitle] Extended Properties.
2. Add the extended properties.
   1. Enter the applicable property name under Name.

      After being added and saved, the name cannot be modified, but you can delete the extended property completely.
   2. Optional: In the Description field, enter a description for the extended property.
   3. If the extended property should allow multiple values, select the Multivalued check box .

> **Note**
>
> If you have initially added a single-valued extended property, you can make it a multivalued extended property by selecting the **Multivalued** check box later.
> If you have initially added a multivalued extended property and you clear the **Multivalued** check box later, when you try to make changes to a connection or an OAuth client that has been configured with multiple values for this extended property, the administrative console prompts you to reconfigure the connection or the client until only one value exists for this extended property.

4. Click Add.

5. Optional: Repeat these steps to define additional extended properties.

Click Edit, Update, or Cancel to make or undo a change to an existing entry. Click Delete or Undelete to remove an existing entry or cancel the removal request.

Log settings

To help you resolve problems, you can use the Log settings window to temporarily enable detailed, or verbose, message logging for specific server log categories.

About this task

PingFederate records runtime and administrative server activities in `<pf_install>/pingfederate/log/server.log`. Enabling verbose logging changes the log level from `INFO` to `DEBUG` or `TRACE`, depending on the log category.

> **Important**
>
> Verbose messages in some categories can include sensitive information. Also, logging verbose messages can decrease server performance. Therefore, when you finish troubleshooting, disable verbose logging.

The following table describes the log categories.

Log category	Description
Core	Debug logging for core components.
Policy Tree	Debug logging for policy trees.
Data Store Response Times	Log response times for data store requests.
Trusted CAs	Log PingFederate and JRE trusted CAs when they are loaded.
XML Signatures	Debug logging for XML signature operations.
HTTP Request Headers	Log HTTP request headers. PingFederate might log sensitive information, such as passwords, when you enable verbose message logging for this category.
HTTP Request Parameters	Log HTTP GET request parameters. PingFederate might log sensitive information, such as passwords, when you enable verbose message logging for this category.

Log category	Description
REST Data Store Requests and Responses	Log REST datastore requests and responses. PingFederate might log sensitive information, such as passwords, when you enable verbose message logging for this category.

> **Note**
>
> You can customize the log categories by editing the configuration file, `<pf_install>/pingfederate/server/default/conf/log4j-categories.xml` . Changes to this file affect both the **Log Settings** window and the `/serverSettings/logSettings` in the administrative API. For more information, see the comments in `log4j-categories.xml` .

To enable verbose logging:

Steps

1. Go to System > Server > Log Settings.
2. Enable Verbose logging for one or more categories.

   By default, no categories have verbose logging enabled.
3. Click Save.
4. If PingFederate is deployed in a cluster, replicate the changes to the other servers in the cluster.

   For more information, see Cluster management.

Related links

- PingFederate log files

Creating a log category

Create a custom log category to help you report specific troubleshooting information in PingFederate.

About this task

Starting with version 11.2, PingFederate allows you to create custom logging categories. These categories allow you to track activities that are available but aren't logged by default.

Steps

1. Open the `<pf_install>/pingfederate/server/default/conf/log4j-categories.xml` file and add a category entry.

   Example:

   ```
   <category id="formAdapter" name="HTML Form Adapter" offLevel="INFO" onLevel="DEBUG"
   description="Enable debug logging for the PingFederate HTML Form adapter."/>
   ```

Category entry parameters

Parameter	Description
`id`	The reference name for the logger. Must be a unique alphanumeric string.
`name`	The friendly name for the logger. The name will appear in the admin console, so it should be descriptive of the logger.
`offLevel`	The default level of logging and verbosity for when the category is disabled. Valid values are: ○ `FATAL` ○ `ERROR` ○ `WARN` ○ `INFO` (recommended) ○ `DEBUG` ○ `TRACE` For more information, see Log4j 2 logging service and configuration.
`onLevel`	The increased level of verbosity for when the category is active. Used to troubleshoot issues. Valid values are: ○ `FATAL` ○ `ERROR` ○ `WARN` ○ `INFO` ○ `DEBUG` (Recommended) ○ `TRACE` For more information, see Log4j 2 logging service and configuration.
`description`	A friendly description for the logger. The description will appear in the admin console.

2. Save and close the `log4j-categories.xml` file.

3. Open the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` and, under the `Loggers` section, add a logger entry.

   Example:

```
<!-- Form Adapter logging -->
<Logger name="com.pingidentity.adapters.htmlform.idp" level="${sys:pf.log.level.formAdapter:-INFO}"/
>
```

   For the example logger entry `sys:pf.log.level.formAdapter:-INFO`, the syntax is as follows:

`sys:pf.log.level`

This is a constant value for all logger entries.

`formAdapter`

The `id` value from the category entry. Case sensitive.

`:-INFO`

The default starting logging level. Should match the `offLevel` value in the category entry.

4. Save and close the `log4j2.xml` file.

5. Copy both files to each PingFederate instance in the cluster that you want the settings to apply to.

6. Restart PingFederate.

Removing the Log Settings window

About this task

You can remove the Log Settings window from the administrative console, requiring administrators to use only the administrative API for enabling verbose messaging. The log settings are located in the administrative API at `/serverSettings/logSettings`.

Steps

1. Go to the `<pf_install>/pingfederate/server/default/data/config-store` directory.

2. Open the `org.sourceid.saml20.domain.mgmt.impl.LogSettingsManagerImpl.xml` file in a text editor.

3. Change the value of the `ShowLogSettingsPage` parameter to `false`.

4. Save your changes.

General settings

The General Settings window lets you change some of the general system-wide settings for your PingFederate servers.

About this task

The settings in this window are independent of one another.

Steps

1. Go to System > Server > General Settings.

2. Change any of the following settings:

   - To turn off automatic multi-connection error checking, select the Disable Automatic Connection Validation check box. This check box is not selected by default.

   - To override the verbosity of runtime transaction logging for all identity provider connections, select an option on the IdP Connection Transaction Logging Override menu. The default setting is Don't Override.

- To override the verbosity of runtime transaction logging for all service provider connections, select an option on the SP Connection Transaction Logging Override menu. The default setting is Don't Override
- To fine tune the caching interval for datastore validation, change the number of seconds in the Data-Store Validation Interval field. A value of `0` turns off the caching and validation tests are executed with each access. The default value is `300` seconds (five minutes). This setting applies to all datastores.

  > **Note**
  >
  > As you configure components on the administrative console, PingFederate performs connectivity tests against the applicable datastores. By default, PingFederate saves successful test results for five minutes. This design improves the performance of the administrative console by reducing the number of calls it makes to the target servers and the amount of time it takes to move from one configuration window to another.

- To let PingFederate use the value of a specific request header for the request ID instead of generating a request ID, enter the name of the request header in the Request Header for Correlation ID field. By default, the field is empty.

  > **Note**
  >
  > When a PingFederate endpoint receives HTTP requests, it records request IDs at the DEBUG level of the server log. When the **Request Header for Correlation ID** field specifies a request header, if a PingFederate endpoint receives a request that includes that header, and if the header's value contains 1 to 50 alphanumeric characters and hyphens, then PingFederate uses that value for the request ID. Otherwise, PingFederate uses a unique value that it generates for the request ID.
  > If the request causes PingFederate to send an LDAP call to PingDirectory, PingFederate includes the request ID in the LDAP call. For auditing and troubleshooting, you can correlate the request ID in the PingFederate log with the request ID in the PingDirectory log. For more information, see Correlating PingFederate events with PingDirectory LDAP activities.

3. Save the changes.

Metadata

You can configure metadata settings, export metadata, or sign metadata XML files.

These menu items are located in the Protocol Metadata window.

Metadata settings

Configure metadata settings using the Pingfederate administrative console.

In the Metadata Settings window, you can configure the contact information to be included in your SAML metadata, and the metadata signing policy for metadata provided by the `/pf/federation_metadata.ping` federation metadata endpoint. You can also configure the validity of manual metadata exports and metadata provided by the metadata endpoint, and the frequency of automatic reloading of SAML metadata from partners.

Entering system information

Provide the contact information to be included in your SAML metadata.

Steps

1. Go to System > Protocol Metadata.

2. In the Metadata Settingswindow, on theSystem Info tab, enter the desired information.

3. Click Next.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Result

PingFederate includes the contact information in the manual metadata exports and metadata provided by the `/pf/federation_metadata.ping` federation metadata endpoint.

Configuring metadata signing

Configure metadata signing using the PingFederate administrative console.

About this task

PingFederate generates publicly available metadata for partners through the federation metadata endpoint, `/pf/federation_metadata.ping`. Although optional, signing the the metadata is recommended so that partners can verify the authenticity of the metadata.

Steps

1. Go to System > Protocol Metadata.

2. In the Metadata Settings window, on theMetadata Signing tab select a certificate from the Signing Certificate list.

   If you have not yet created or imported your certificate into PingFederate, click Manage Certificates and use the Certificate Management configuration wizard to complete the task.

3. Optional: Select a signing algorithm from the list.

   The default selection is RSA SHA256 or ECDSA SHA256 depending on the key algorithm of the chosen signing certificate. Make a different selection if you and your connection partner have agreed to use a stronger algorithm. The following table shows the URIs of the supported algorithms. For a list of the available signing algorithms and their URIs, see Signing algorithms.

   The public key of the metadata signing certificate is included as part of the metadata.

4. Click Next.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring metadata lifetime

Configure the lifetime of metadata to optimize cache performance in your environment.

About this task

PingFederate provides metadata for SAML and WS-Federation connections and supports automatic update for SAML connections by reloading metadata URLs provided by the partners

Metadata publication

PingFederate includes expiration information in metadata. It indicates to partners whether they have reasonably up-to-date information about your server.

Metadata consumption

PingFederate supports automatic reloading of metadata by URL for SAML connections.

Steps

1. Go to System > Protocol Metadata.
2. In theMetadata Settings window, on the Metadata Lifetime tab, configure your metadata.

Option	Action
Adjust the validity of your metadata	Modify the Cache Duration field value, in minutes. The default value is `1440` or 1 day.
Adjust the frequency of automatic reloading of SAML metadata	Modify the Reload Delay field value, in minutes. The default value is `1440` or 1 day.

3. Click Next.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers you the opportunity to do so.

Reviewing metadata settings

Review your metadata settings before completing configuration.

Steps

1.

2. To amend your configuration, click the corresponding tab title and then follow the steps to complete the task.
    - To keep your changes, click Save.
    - To discard your changes, click Cancel.

Metadata export

Use the Pingfederate administrative console to export metadata.

The SAML standards define a metadata exchange schema for conveying XML-formatted information between two SAML entities. Metadata includes endpoint URLs, binding types, attributes, and security-policy information, which helps federation partners expedite their configurations.

In the Metadata Export window, you can export metadata to an XML file by selecting any SAML browser single sign-on (SSO) connection or specifying the desired information manually. The former is also available as a per-connection action item on the Connections window. The latter addresses the scenarios where you have not yet created a SAML connection, or you want to generate one SAML metadata XML file for multiple partners.

To export a SAML metadata file, select the role your PingFederate server plays, configure the export options and metadata signing policy, and then save the SAML metadata to an XML file.

For more information on exporting metadata for any SAML Browser SSO connection to an XML file, see Exporting connection-specific SAML metadata.

For more information on manually selecting specific information and exporting a metadata XML file, see Exporting selected SAML metadata.

Exporting connection-specific SAML metadata

You can export metadata for any SAML browser single sign-on (SSO) connection to an XML file.

About this task

This is useful in a situation where you have already created a SAML browser SSO connection to your partner and the partner prefers consuming SAML metadata by file.

Steps

1. Go to System > Protocol Metadata > Metadata Export.
2. On the Metadata Role tab, select the applicable role, and click Next.
3. On the Metadata Mode tab, select the Use a connection for metadata generation option.

    If the secondary HTTPS port is configured and you want to use it for the SOAP channel, select the Use the secondary port for SOAP channel check box.

    > **Note**
    >
    > If certificate-based authentication is configured for the SOAP channel, you must configure the `pf.secondary.https.port` property in the `<pf_install>/pingfederate/bin/run.properties` file and select this check box.

4. On the Connection Metadata tab, select the applicable SAML browser SSO connection from the list.

   Choose from:

   - Virtual Server ID

     If the selected connection contains two or more virtual server IDs, you must select the virtual server ID that you want to use during the export. The protocol endpoints in the metadata file are specific to the selected virtual server ID. If you decide to update the virtual server ID at a later time, re-export the connection metadata for your partners

   - Virtual Host Name

     If PingFederate is configured with one of more virtual server host names, you can select the applicable virtual host name from the list. If a selection is made, PingFederate use that virtual host name when generating the metadata file. If left blank, PingFederate uses its base URL in the metadata file. If you decide to update one or more virtual host names at a later time, re-export the connection metadata for your partners.

     On the Metadata Signing tab, select a certificate to use for signing the metadata XML file.Select a certificate from the Signing Certificate list.

     If you have not yet created or imported your certificate into PingFederate, click Manage Certificates and use the Certificate Management configuration wizard to complete the task.Select the related check boxes to include the public key information and the raw key in the signed XML file.Select a signing algorithm from the list.

     The default selection is RSA SHA256 or ECDSA SHA256, depending on the key algorithm of the chosen signing certificate. Make a different selection if you and your connection partner have agreed to use a stronger algorithm. For a list of the available signing algorithms and their URIs, see Signing algorithms.

5. On the Export & Summary tab, click Export to save the metadata XML file, then click Done.

6. Pass the metadata XML file to your partner.

Exporting selected SAML metadata

You can manually select the desired information and export a metadata XML file.

About this task

This type of export is useful for the following situations:

- You have not yet created a SAML browser single sign-on (SSO) connection to the partner but would like to help your partner with its configuration by including selected information in a metadata XML file.
- You want to export a SAML metadata with selected information, which can be passed to multiple partners to expedite their configurations.

Steps

1. Go to System > Protocol Metadata > Metadata Export.

2. On the Metadata Role tab, select the applicable role.

3. On the Metadata Mode tab, select the Select information to include in metadata manually option.

If the secondary HTTPS port is configured and you want to use it for the SOAP channel, select the Use the secondary port for SOAP channel check box.

> ⓘ **Note**
>
> If certificate-based authentication is configured for the SOAP channel, you must configure the `pf.secondary.https.port` property in the `<pf_install>/pingfederate/bin/run.properties` file and select this check box.

4. On the Protocol tab, select the desired version of the SAML protocol from the list.
5. On the Virtual Host Name tab, select the applicable virtual host name from the list.

   This tab is shown and applicable only if PingFederate is configured with one of more virtual server host names.

   If a selection is made, PingFederate use that virtual host name when generating the metadata file. If left blank, PingFederate uses its base URL in the metadata file. If you decide to update one or more virtual host names at a later time, re-export the connection metadata for your partners.
6. Optional: On the Attribute Contract tab, you can perform the following actions.

Action	Description
Add	Add an attribute contract by entering the contract's name and clicking Add.
Edit	Modify an existing attribute contract by clicking Edit. To save your change, click Update. To cancel your change, click Cancel.
Delete	Delete an existing attribute contract by clicking Delete.

7. On the Signing Key tab, do the following:
   1. Optional: If you want to include a public key that this system uses for digital signatures, select a key from the Digital Signature Keys/Certs list.

      If you have not yet created or imported a digital signature key to PingFederate, click Manage Certificates and use the Digital Signature Settings wizard to complete the task.
   2. Optional: If you want to give partners an alternative key, select another key from the Secondary Digital Signature Keys/Certs list.

      > ⓘ **Note**
      >
      > You can't configure a secondary signing certificate if the primary certificate has certificate rotation enabled. You also can't use a certificate that has rotation enabled as the secondary signing certificate.

   On the Metadata Signing tab, select a certificate to use for signing the metadata XML file.Select a certificate from the Signing Certificate list.

If you have not yet created or imported your certificate into PingFederate, click Manage Certificates and use the Certificate Management configuration wizard to complete the task.Select the related check boxes to include the public key information and the raw key in the signed XML file.Select a signing algorithm from the list.

The default selection is RSA SHA256 or ECDSA SHA256, depending on the key algorithm of the chosen signing certificate. Make a different selection if you and your connection partner have agreed to use a stronger algorithm. For a list of the available signing algorithms and their URIs, see Signing algorithms.

8. Optional: On the XML Encryption Certificate tab, select the certificate that your partner can use to encrypt XML content.

   Applicable only when you have selected SAML 2.0 on the Protocol tab.

   If you have not created or imported your certificate into PingFederate, click Manage Certificates and use the Certificate Management configuration wizard to complete the task.

9. On the Export & Summary tab, click Export to save the metadata XML file, then click Done.

10. Pass the metadata XML file to your partner or partners.

File signing

Applying a digital signature to an XML file assures the authenticity and integrity of the original source.

If you have previously exported an unsigned metadata XML file, you can create a new signed metadata XML file based on the unsigned metadata in the File Signing window.

Signing XML files

Use the Pingfederate administrative console to sign XML files.

Steps

1. Go to System > Protocol Metadata > [.wintitle] File Signing.

2. On the Select Metadata File tab, choose your metadata file.

3. On the Digital Signature Settings tab, select a signing certificate from the list.

   If you have not yet created or imported your certificate into PingFederate, click Manage Certificates and use the Certificate Management configuration wizard to complete the task.

   1. Clear the related check boxes to exclude the public key information and the raw key from the signed XML file.

   2. Select a signing algorithm from the list.

   The default selection is RSA SHA256 or ECDSA SHA256, depending on the key algorithm of the chosen signing certificate. Make a different selection if you and your connection partner have agreed to use a stronger algorithm. For a list of the available signing algorithms and their URIs, see Signing algorithms.

4. On the Export & Summary tab, click Export to save the digitally signed file.

5. Click Done.

Monitoring and notifications

You can set up notifications for licensing events, certificate events, and SAML metadata update events. These menu items are located in System > Monitoring & Notifications.

Runtime notifications

You can configure PingFederate to generate notification messages for various events.

Complete this configuration on the Runtime Notifications window.

Licensing events

Depending on your licensing agreement, your PingFederate license might have an expiration date. You can configure PingFederate to generate notification messages when your license is about to expire. This option does not appear when you have a perpetual license.

Certificate events

When enabled, PingFederate can generate notification messages when a certificate is about to expire or has expired. If you have also configured PingFederate to rotate self-signed certificates, PingFederate can generate notification messages after generating a certificate and after its activation.

SAML metadata update events

If you have enabled automatic reloading of partner metadata in any SAML browser single sign-on (SSO) connections, you can configure PingFederate to generate notification messages when it detects changes after pulling the metadata from the partners.

> **Note**
>
> In a clustered PingFederate environment, notification messages for licensing and certificate events are generated by only one of the nodes. If that node leaves the cluster, planned or not, another node picks up this task automatically.

Regardless of runtime notification settings, PingFederate always records license expiration events, certificate events, and SAML metadata update events in the server log. For more information, see PingFederate log files.

Configuring runtime notifications

Use the PingFederate administrative console to configure and manage runtime notifications. These notifications provide alerts about license status, ensuring uninterrupted server functionality. They also notify about critical certificate events, such as expiration or changes, enabling users to maintain secure communication. Additionally, notifications for SAML metadata updates help users adapt their configurations and ensure smooth authentication and authorization processes. Overall, these runtime notifications ensure compliance, security, and uninterrupted services for users.

Steps

1. Go to System > Monitoring & Notifications > [.wintitle] Runtime Notifications**.

2. Optional: Configure licensing events:

    1. Select the Notification for Servers Licensing Events check box.

        > **Note**
        >
        > This check box appears only if your PingFederate license has an expiration date.

    2. Enter the email address of the intended recipient.

    3. Select a notification publisher instance from the list.

    If you have not yet configured the desired notification publisher instance, click Manage Notification Publishers.

3. Optional: Configure certificate events:

    1. Select the Notification for Certificate Events check box.

    2. Optional: Enter the initial warning time period (in days) in the Initial Warning Event field.

    3. Enter the final warning time period (in days) in the Final Warning Event field.

    4. Select a notification publisher instance from the list.

    Choose from:

    - To send certificate events by email, select Default and enter the email address of the intended recipient in the Email Address field that displays below.

    - To record runtime certificate advance warnings in the `server.log` file, select Logging Only.

        > **Note**
        >
        > If you select this option, you are not required to set any notification publisher.

4. Optional: Configure SAML metadata update events:

    1. Select the Notification for SAML Metadata Update Events check box.

    2. Enter the email address of the intended recipient.

    3. Select a notification publisher instance from the list.

    If you have not yet configured the desired notification publisher instance, click Manage Notification Publishers.

5. Configure thread pool exhaustion events:

    1. Select the Notification for Thread Pool Exhaustion Events check box.

    2. Select the Thread Dump check box to enable PingFederate to dump threads in the event of a thread pool exhaustion.

Related links

- Managing notification publisher instances

Datastores

Datastores represent external systems where user attributes and other data are stored. Once defined, you can configure PingFederate to retrieve user attributes from datastores for contract fulfillment and token authorization in various use cases.

To manage datastores, go to System > Data & Credential Stores > Data Stores.

- To create a new datastore, click Add New Data Store and then follow the configuration wizard to complete the task.
- To modify an existing datastore, select the datastore and then follow the configuration wizard to complete the task.
- To review usage of an existing datastore, click Check Usage under Action.
- To remove an existing datastore or cancel the removal request, click Delete or Undelete under Action.

> **Note**
>
> You can only remove datastores that are not currently in use.

- To fine-tune the caching interval for datastore validation, update the Data-Store Validation Interval field value to the desired amount of time in seconds.

> **Note**
>
> As you configure various components on the administrative console, PingFederate performs connectivity tests against the applicable datastores. By default, PingFederate stores successful test results for five minutes. This design improves the performance of the administrative console by reducing the number of calls it makes to the target servers and the amount of time it takes to move from one configuration window to another.

> **Note**
>
> The default value is `300` seconds (five minutes). A value of `0` turns off the caching and validation tests are executed with each access. This setting applies to all datastores.

Adding a new datastore

On the Data Stores window, you can create and configure a new datastore.

Steps

1. Go to System > Data & Credential Stores > Data Stores.
2. Click Add New Data Store.
3. Enter a name for the datastore.
4. From the Type list, select the type of datastore.

   Available types are limited to the ones currently installed on your server.
5. Optional: To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log check box.

6. Click Next.

Configuring a PingOne LDAP Gateway datastore

The PingOne LDAP Gateway reduces the complexity of moving to the cloud while maintaining connectivity to on-premise end-user data.

Before you begin

Make sure you have the following in place:

- A PingOne environment configured with an LDAP gateway. Learn more in Gateways and Adding a LDAP Gateway.
- A connection between PingFederate and PingOne. Learn more in Creating connections to PingOne.

About this task

When PingFederate is deployed off-premise as a PingOne Advanced Service or in your own cloud deployment, you can configure the PingOne LDAP Gateway datastore to enable PingFederate to access an on-premise LDAP directory for HTML Form Adapter functionality, provisioning, customer identity access management (CIAM), and other areas.

> **Note**
>
> Currently, you cannot use the PingOne LDAP Gateway for grant storage, persistent authentication sessions, and OAuth client records. All other LDAP datastore functionality works in the same way as the direct LDAP datastore.

Steps

1. Go to System > Data & Credential Stores > Data Stores.
2. In the Data Stores page, click Add New Data Store.
3. On the Data Store Type tab, enter a name for the datastore in the Name field.
4. In the Type list, select PingOne LDAP Gateway.
5. (Optional) To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log checkbox.
6. Click Next.
7. In the LDAP Gateway Configuration page, configure your LDAP Gateway as follows.
    1. In the PingOne Environment list, select your PingOne environment.
    2. In the PingOne LDAP Gateway list, select your PingOne LDAP gateway.
8. Click Test Connection to determine whether the administrative node can communicate with the specified datastore.

> **Note**
>
> - Datastore validation is not enabled during configuration, which lets you configure datastores without requiring a successful connection between the administrative node and the datastore. You can also save the datastore even if the connection is not currently successful.
> - Due to the implementation of Client TLS Certificate Authentication in Active Directory, when the **LDAP Type** is **Active Directory** and the **Authentication Method** is **Client TLS Certificate**, the connection test always succeeds, even when an incorrect certificate is selected. This is not the case when PingFederate attempts to retrieve data from the datastore because the connection will fail to bind.

9. Click Advanced to configure LDAP attributes to be handled as binary data.
10. Click Next to view the summary of your LDAP gateway datastore configuration.
11. Click Save.

Configuring a JDBC connection

You can establish a Java Database Connectivity (JDBC) connection to your database server. JDBC supports connections to relational databases, such as Microsoft SQL Server, Azure SQL Managed Instance, Oracle MySQL, or PostgreSQL.

About this task

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers.

Steps

1. Go to System > Data & Credential Stores > Data Stores.
2. On the Data Stores window, click Add New Data Store.
3. On the Data Store Type tab, type a name for the datastore.
4. From the Type list, select Database (JDBC). Click Next.
5. Optional: To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log check box.
6. Click Next.
7. In the Database Config window, configure your JDBC connection. Information about each field is provided in the following table.

Field	Description
JDBC URL	The location of the database server and the database. The structure of the JDBC URL varies depending on the vendor. You can add multiple JDBC URLs. You can also specify which node is the default by clicking Set as Default under Action. For regional deployments, specify region-specific URLs on different rows, along with the node tags for the region. Tag the nodes that will use the URL with the `node.tags` property in the `run.properties` file. Failover may be supported within a single region via driver-specific URL parameters. Failover is not supported across different rows. **Tip** For Oracle MySQL, to enable automatic reconnection attempts when the connection is not available at runtime, enter a SQL statement in the **Validate Connection SQL** field and add the following query string to the JDBC URL: `?autoReconnect=true`
Tags	Tags are defined in the `node.tags` property in the `<pf_install>/pingfederate/bin/run.properties` file. For a description of the `node.tags` property, see Deploying cluster servers. In PingFederate deployments that are regional, you can enter one or more tags for a JDBC URL, which specifies with which datastore that particular PingFederate node should communicate. If none of the tags match what is defined for the `node.tags` property, the default node is used. The following rules apply to tags: ◦ You must separate multiple tags specified for one node with spaces. ◦ You can't use a tag more than once per datastore. ◦ Tags are optional. If needed, you can configure a non-default node without tags. Doing this is useful if you are not yet ready to tag the node, or if you are still in the planning stage but want to enter the address for the node now.
Driver Class	The name of the driver class used to communicate with the source database. The driver class name should be supplied by the database software vendor in a JAR file.

Field	Description
Username	The name that identifies the user when connecting to the database. **Note** Leave this field blank if no credentials are needed when authenticating to Azure SQL Managed Instance through Azure Active Directory and a JDBC connection string URL.
Password	The password needed to access the database. **Note** Leave this field blank if no credentials are needed when authenticating to Azure SQL Managed Instance through Azure Active Directory and a JDBC connection string URL.
Validate Connection SQL (Optional but recommended)	A simple SQL statement used by the PingFederate runtime server to verify that the database connection is still active and to reconnect if needed. If a SQL statement is not provided here, PingFederate might not reconnect to the database if the connection is broken. **Important** Ensure that the SQL statement is valid for your database. For example: ◦ `SELECT 1 from dual` (for Oracle Database or Oracle MySQL) ◦ `SELECT getdate()` (for Microsoft SQL Server or Microsoft Azure SQL Managed Instance) ◦ `SELECT 1` (for PostgreSQL) **Tip** To use this feature for Oracle MySQL, you must also add the `?autoReconnect=true` query parameter to the JDBC URL.
Mask Values in Log	Determines whether all attribute values returned through this datastore should be masked in PingFederate logs. Applicable only when editing an existing datastore.

Field	Description
Allow Multi-Value Attributes	When selected, indicates that the JDBC datastore can select more than one record from a column and return the results as a multivalued attribute. Otherwise, a query returns only the first value in the column.

Click Test Connection to determine whether the administrative node can communicate with the specified datastore.

> **Note**
>
> - Datastore validation is not enabled during configuration, which lets you configure datastores without requiring a successful connection between the administrative node and the datastore. You can also save the datastore even if the connection is not currently successful.
> - Due to the implementation of Client TLS Certificate Authentication in Active Directory, when the **LDAP Type** is **Active Directory** and the **Authentication Method** is **Client TLS Certificate**, the connection test always succeeds, even when an incorrect certificate is selected. This is not the case when PingFederate attempts to retrieve data from the datastore because the connection will fail to bind.

8. Click Advanced to configure additional settings.

    1. On the Advanced Database Options window, click Apply Defaults to view or restore default values.

        > **Tip**
        >
        > The default values are conservative based on the server thread pool settings configured in the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file. If any changes are made to thread pooling, we recommend updating settings as outlined in the next step.

    2. Configure advanced settings.

        For more information about each field, see the following table.

Field	Description
Minimum Pool Size	The smallest number of database connections that can remain in the pool for the given datastore. A minimum value of `0` means that the minimum number of connections in the pool is zero. **Note** For optimal performance, the value for this setting should equal 50% of the `maxThreads` value in the Jetty server configuration (see Configuring connection pools to datastores). Note that PingFederate does not establish the connection pool for the given datastore until it receives a request that requires one or more attributes from that datastore. The default value (after clicking on Apply Defaults)is `10`.

Field	Description
Maximum Pool Size	The largest number of database connections that can remain in the pool for the given datastore. ⓘ **Note** For optimal performance, the value for this setting should equal 75% to 100% of the `maxThreads` value in the Jetty server configuration (see Configuring connection pools to datastores). The default value (after clicking on Apply Defaults)is `100`.
Blocking Timeout (ms)	The amount of time a request waits to get a connection from the connection pool before it fails. A value of `-1` means that a request waits indefinitely for the connection pool to return a connection. The default value (after clicking on Apply Defaults) is `5000`.
Idle Timeout (min)	The length of time the connections can sit idle in the pool before it closes them. A value of `-1` means that the connection pool does not close its connections (once established). Note that PingFederate maintains the minimum connection pool for the given datastore once the pool is established. The default value (after clicking on Apply Defaults)is `5`.

9. Click Save to save your configuration.

Configuring an LDAP connection

In the Data Stores configuration window, establish an Lightweight Directory Access Protocol (LDAP) connection to your directory server.

Steps

1. Go to System > Data & Credential Stores > Data Stores.

2. On the Data Stores window, click Add New Data Store.

3. On the Data Store Type tab, type a name for the datastore.

4. From the Type list, select Directory (LDAP).

5. Optional: To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log check box.

6. Click Next.

7. On the LDAP Configuration tab, configure your LDAP connection as described in the following table.

Field	Description
Data Store Name	The name of the datastore. This field is visible only when editing an existing datastore.
Hostname(s) (Required)	The network address of the directory server, either an IP address, a host name, or a fully qualified domain name. The entry might include a port number; for example, `10.10.10.101:1389`. For failover, enter multiple directory servers, each separated by a space. In addition to network error conditions, PingFederate also fails over to the next server if the current server returns an LDAP system error. ⓘ **Note** If multiple directory servers are specified, each server must be accessible by using the same user distinguished name (DN) and password (unless the **Bind Anonymously** check box is selected). You can add multiple hostnames. You can also specify which node is the default by clicking Set as Default under Action. PingFederate can also leverage DNS service records to locate the directory server (when the Use DNS SRV Record check box is selected), in which case the value of this field must be a single domain; for example, `example.com`.
Tags	Tags are defined in the `node.tags` property in the `<pf_install>/pingfederate/bin/run.properties` file. See Deploying cluster servers for a description of the `node.tags` property. In regional PingFederate deployments, you can enter one or more tags for a host name, which specify with which datastore that particular PingFederate node should communicate. If none of the tags match what is defined for the `node.tags` property, the default node is used. The following rules apply to tags: ◦ You must separate multiple tags specified for one node with spaces. ◦ You cannot use a tag more than once per datastore. ◦ Tags are optional. If needed, you can configure a non-default node without tags. This is useful if you are not yet ready to tag the node, or if you are still in the planning stage but want to enter the address for the node now.
Use LDAPS	When selected, PingFederate connects to the directory server using LDAPS. This selection applies equally to all servers specified in the Hostname(s) field. ◇ **Important** You should secure all LDAP connections by using LDAPS. ⓘ **Note** To enable the password changes, password reset, or account unlock features in the HTML Form Adapter against Microsoft Active Directory, you must secure the connection to your directory server using LDAPS; Microsoft Active Directory requires this level of security to allow password changes. This check box is cleared by default.

Field	Description
Use DNS SRV Record	Used in conjunction with the domain information defined in the Hostname(s) field and the preference of LDAP or LDAPS, PingFederate uses DNS SRV records to locate the directory server when this check box is selected. You can fine-tune the TTL value and the record prefixes on the Advanced LDAP Options window. **Note** When the DNS returns multiple SRV records, PingFederate uses the record with the lowest-numbered priority value and fails over to the record with the next lowest priority value. If multiple records share the same priority value, PingFederate uses the records with the highest-numbered weight value. PingFederate repeats this exercise until it establishes a connection or fails to connect to any directory server after taking all records into consideration. This check box is cleared by default.
Follow LDAP Referrals	Select this check box to let the datastore follow LDAP referrals on Microsoft Active Directory or Oracle Unified Directory. **Note** PingFederate always follows LDAP referrals from PingDirectory based on the recommended PingDirectory configuration.
LDAP Type (Required)	If you are using this datastore for outbound provisioning and your directory server is PingDirectory, Microsoft Active Directory or Oracle Unified Directory, select the applicable type from the list, such that PingFederate can pre-populate many provisioning settings on Outbound Provisioning > Channel > Source Settings. **Tip** If your directory server is not PingDirectory, Microsoft Active Directory, or Oracle Unified Directory, you can define a custom LDAP Type to streamline the outbound provisioning configuration. The LDAP type is also used to enable password-change messaging between Microsoft Active Directory and PingFederate when an HTML Form Adapter instance is used.

Field	Description
Authentication Method	Select how PingFederate will authenticate with the directory server, which depends on the configuration of the directory server: ◦ None (Anonymous): Select this option if your directory server supports anonymous binding and no credentials are needed to access the directory server. **Tip** For inbound provisioning, because PingFederate needs to manage local user records, your directory server might require a specific service account to handle the communication between PingFederate and the target directory server. If you choose an anonymous binding, ensure that this access level provides permission to search the directory for user-account information. ◦ Simple: Select this option if the directory server requires PingFederate to provide a user domain name (DN) and password to authenticate. After selecting this option, select a Credential Storage option: Internally Managed or Secret Manager. Then specify the User DN and either the Password or Password Reference. [.uicontrol]Client TLS Certificate: This option is available if you selected [.uicontrol]Use LDAPS. Select this option when using mutual TLS (mTLS) authentication. PingFederate authenticates by presenting the client transport layer security (TLS) certificate that you select in the [.uicontrol]Client Certificate** list.
Credential Storage	Select whether PingFederate with store the credentials internally or in a secret manager. For more information, see Secret managers. These settings are visible only when Authentication Method is set to Simple.
User DN	The user name credential required to access the directory server. This field is visible only when Authentication Method is set to Simple. **Important** The service account must have permission to search the directory for user-account information. If your use cases involve reading from the directory server without creating, updating, or deleting any records, consider using a service account with read-only access. For inbound provisioning, a service account with permission to create, read, update, and delete users and groups is required. When connecting to a Microsoft Active Directory server, enter a Microsoft Active Directory user account. Do not use a computer account. When connecting to PingDirectory or Oracle Unified Directory, configure proxied authorization for the service account on the directory server if you intend to enable self-service password reset in any HTML Form Adapter instances that use this datastore. For more information, see Proxied authorization.
Password	The credential required to access the directory server for simple authentication. This field is visible only when Authentication Method is set to Simple and Credential Storage is set to Internally Managed.

Field	Description
Password Reference	The reference code PingFederate uses to retrieve the password it needs to access the directory server for simple authentication. This field is visible only when Authentication Method is set to Simple and Credential Storage is set to Secret Manager. For information about generating a password reference code, see Using passwords in secret managers to access datastores.
Client Certificate	Select the client TLS certificate that PingFederate will present to the directory server for authentication. This list is visible only when Authentication Method is set to Client TLS Certificate. **Tip** If you have not yet created or imported a client certificate, click the **Manage SSL Client Keys & Certificates** button to do so. For more information, see Manage SSL client keys and certificates.
Mask Values in Log	Determines whether all attribute values returned through this datastore should be masked in PingFederate logs. This check box is visible only when editing an existing datastore.

8. Click Test Connection to determine whether the administrative node can communicate with the specified datastore.

> **Note**
>
> - Datastore validation is not enabled during configuration, which lets you configure datastores without requiring a successful connection between the administrative node and the datastore. You can also save the datastore even if the connection is not currently successful.
> - Due to the implementation of Client TLS Certificate Authentication in Active Directory, when the **LDAP Type** is **Active Directory** and the **Authentication Method** is **Client TLS Certificate**, the connection test always succeeds, even when an incorrect certificate is selected. This is not the case when PingFederate attempts to retrieve data from the datastore because the connection will fail to bind.

9. Optional: Click Advanced. If you choose an anonymous binding, configure additional settings in the Advanced LDAP Options window Click Save.

   Result:

   You are directed back to the LDAP Configuration tab.

10. On the Summary tab, click Save.

Related links

- Setting advanced LDAP options

Setting advanced LDAP options

PingFederate lets you customize the default settings of both the search pool and the bind pool for each LDAP datastore.

About this task

PingFederate maintains a search pool and a bind pool for each LDAP datastore for optimal performance. The search pool is for LDAP directory searches. The bind pool is for LDAP bind authentication purposes. Use the Advanced LDAP Options page to change default pool settings. These settings are applicable to both the search pool and the bind pool.

When configuring PingFederate to locate the directory server based on DNS SRV record, you can fine-tune the TTL value and the SRV record prefixes.

On the LDAP Binary Attributes tab, can also specify attributes that have values PingFederate must handle as binary data for use in attribute contract fulfillment. Binary attributes are typically used for certificates and images.

Note

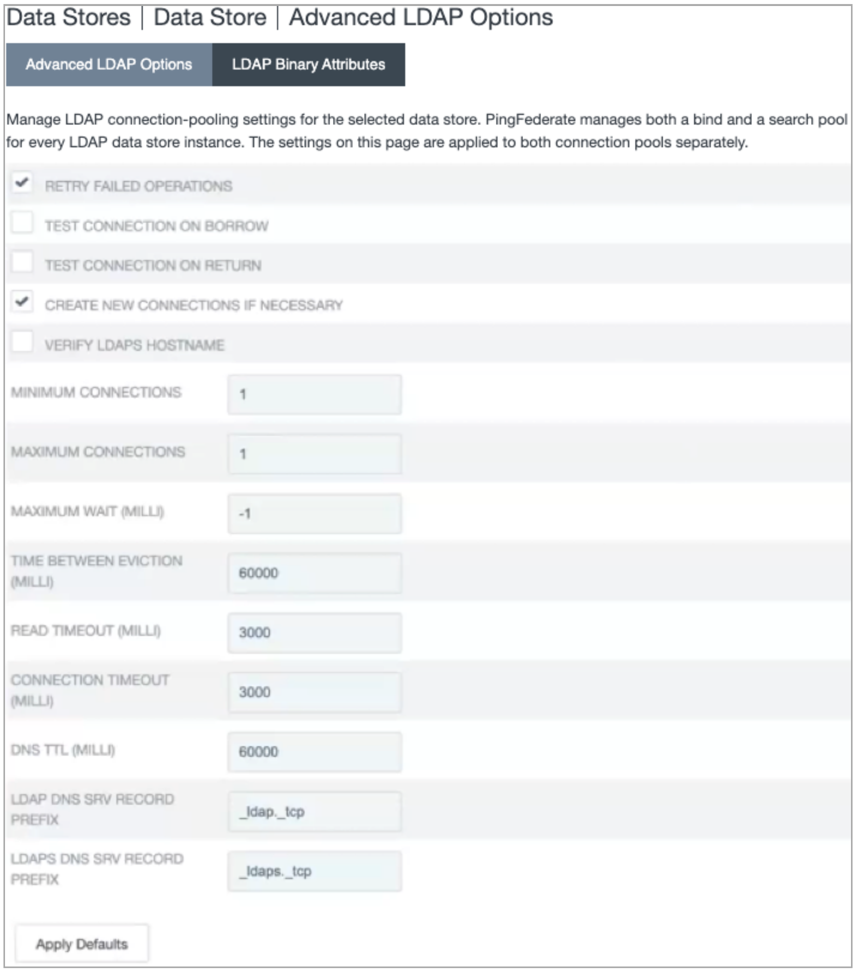
You cannot use binary data in an assertion. You must apply and handle encoding on a per-connection basis. When binary attributes are selected for attribute mapping, the administrative console prompts you to select an encoding type for each binary attribute.

Steps

1. On the Data Store page's LDAP Configuration tab, click Advanced.

   Result:

   The Advanced LDAP Options page opens.



Manage LDAP connection-pooling settings for the selected data store. PingFederate manages both a bind and a search pool for every LDAP data store instance. The settings on this page are applied to both connection pools separately.

☑ RETRY FAILED OPERATIONS

□ TEST CONNECTION ON BORROW

□ TEST CONNECTION ON RETURN

☑ CREATE NEW CONNECTIONS IF NECESSARY

□ VERIFY LDAPS HOSTNAME

MINIMUM CONNECTIONS	1
MAXIMUM CONNECTIONS	1
MAXIMUM WAIT (MILLI)	-1
TIME BETWEEN EVICTION (MILLI)	60000
READ TIMEOUT (MILLI)	3000
CONNECTION TIMEOUT (MILLI)	3000
DNS TTL (MILLI)	60000
LDAP DNS SRV RECORD PREFIX	_ldap._tcp
LDAPS DNS SRV RECORD PREFIX	_ldaps._tcp

2. Optional: To view or restore default values, click Apply Defaults on the Advanced LDAP Options tab.

   The default values are conservative based on the server thread pool settings configured in the `<pf_install>/pingfederate/etc/run.properties` file. If any changes are made to thread pooling, update the settings as outlined in the following step.

3. Configure the advanced settings. For more information about each field, see the following table.

Field	Description
Retry Failed Operations	PingFederate initiates a single retry if a request fails and it appears the connection might have become invalid. The connection is discarded, and PingFederate establishes a new one for the retry. The standard failover logic applies when creating the new connection if failover is enabled. ⓘ **Note** In PingFederate, only operations that do not modify entries (`BIND` , `SEARCH` , and `COMPARE`) are eligible for retry. This checkbox is not selected by default.
Test Connection on Borrow	Indicates whether to validate objects before they are borrowed from the pool. This checkbox is not selected by default.
Test Connection on Return	Indicates whether to validate objects before they return to the pool. This checkbox is not selected by default.
Create New Connection If Necessary	Indicates whether you can create temporary connections when the Maximum Connections threshold is reached. Temporary connections are managed automatically. ⓘ **Note** If disabled, when the **Maximum Connections** value is reached, subsequent requests relying on this LDAP datastore instance might fail. This checkbox is selected by defaul

Field	Description
Verify LDAPS Hostname	Indicates whether to verify that the host name of the directory server matches the subject (CN) or one of the subject alternative names (SANs) from the certificate. **Important** Verify the LDAPS host name for all LDAPS connections. This checkbox is selected by default.
Minimum Connections (Required)	The smallest number of connections that can remain in each pool. A minimum value of `1` creates two connections, one connection in the search pool and one connection in the bind pool. The default value is `10`. **Note** For optimal performance, the value for this setting should equal 50% of the `maxThreads` value in the Jetty server configuration. Learn more in Configuring connection pools to datastores. **Note** PingFederate does not establish the connection pool for the given datastore until it receives a request that requires one or more attributes from that datastore.
Maximum Connections (Required)	The largest number of active connections that can remain in each pool (not including the temporary connections that are managed automatically when the Create New Connection If Necessary checkbox is selected). The value must exceed or equal the Minimum Connections value. **Note** For optimal performance, the value for this setting should equal 75% to 100% of the `maxThreads` value in the Jetty server configuration. Learn more in Configuring connection pools to datastores. The default value is `100`.

Field	Description
Maximum Wait (Milli) (Required)	The maximum number of milliseconds the pool waits for an available connection when trying to obtain a connection from the pool. A value of `-1` causes the pool not to wait at all and to either create a new connection or produce an error (when no connections are available). The default value is `-1`.
Time Between Eviction (Milli) (Required)	The number of milliseconds between periodic background health checks against the available connections in this pool. A value of `-1` disables the evictor. The default value is `60000`.
Read Timeout (Milli) (Required)	The maximum number of milliseconds a connection waits for a response to return before producing an error. A value of `-1` causes the connection to wait indefinitely. The default value is `3000`.
Connection Timeout (Milli) (Required)	The maximum number of milliseconds that a connection attempt can continue before returning an error. A value of `-1` causes the pool to wait indefinitely. The default value is `3000`.
DNS TTL (Milli) (Required)	The amount of time in milliseconds that a previously obtained DNS SRV record remains valid. When this threshold is reached, PingFederate contacts the DNS for a new SRV record to locate the directory server. The default value is `60000`.
LDAP DNS SRV Record prefix (Required)	The prefix that PingFederate uses in its DNS queries for SRV records to locate an LDAP-capable directory server. The default value is `_ldap._tcp`.
LDAPS DNS SRV Record prefix (Required)	The prefix that PingFederate uses in its DNS queries for SRV records to locate an LDAPS-capable directory server. The default value is `_ldaps._tcp`.

4. Optional: To specify LDAP binary attributes:

    1. Click Next on the Advanced LDAP Options tab.

    2. On the LDAP Binary Attributes tab, add, edit, or remove binary attributes.

5. Click Save.

Proxied authorization

When connecting to PingDirectory or Oracle Directory Server, configure proxied authorization for the service account on the directory server if you intend to enable self-service password reset in any HTML Form Adapter instances that use this datastore.

By configuring proxied authorization for the service account on the directory server, users are not allowed to reset their passwords if their accounts are disabled or if they were not granted permission to change their passwords.

For information on configuring proxied authorization for service accounts, see the following table.

Directory server	Reference
PingDirectory	Learn more in Working with Proxied Authorization in the PingDirectory Administration Guide.
Oracle Directory Server	Learn more in Proxy Authorization in the Oracle Fusion Middleware Deployment Planning Guide.
Oracle Unified Directory	Go to Oracle’s online guide Administering Oracle Unified Directory and search for "proxied authorization control" in its glossary.

> **Note**
>
> Microsoft Active Directory does not support proxied authorization. See 3.1.1.3.4.2.4 LDAP_SERVER_WHO_AM_I_OID.

For general information about proxied authorization, see RFC 4370.

Allowing PingFederate to unlock PingDirectory accounts

When connecting to PingDirectory, you can give the service account access to specific attributes that PingFederate reads or modifies when unlocking user accounts.

Steps

1. Create an LDIF file to capture the following ACI information.

   OID

      Step 1.3.6.1.4.1.42.2.27.8.1.17

   Name

      pwdAccountLockedTime

   Permission

      all

      Example:

For more information, see the following example file named `aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="ds-pwp-auth-failure||pwdAccountLockedTime")(version 3.0; acl "Allow unlock
admin to lock and unlock user accounts"; allow (all) userdn="ldap:///
uid=ServiceAccount,ou=Applications,dc=example,dc=com";)
```

2. Use the `ldapmodify` command to configure the required ACI.

   Example:

   ```
   $ ldapmodify -f  <path>/aci.ldif
   -h  <host name>
   -p  <LDAP port>
   -D  <LDAP bind username>
   -w  <LDAP bind password>
   ```

   > **Note**
   >
   > Line breaks are inserted for readability only.

Related links

- Managing Access Control in the PingDirectory Administration Guide

Configuring the password validation details request control ACI

When connecting to PingDirectory, configure the password validation details request control Access Control Instruction (ACI) to provide user-friendly messages when users fail to change or reset their passwords through the self-service account management capabilities in any HTML Form Adapter instances that use the datastore.

About this task

For self-service password management, where the user knows the current password and wants to update it, the service account of the datastore must have the password validation details request control ACI. For self-service account recovery, where the user wants to define a new password after forgetting the current password, the user account needs the same ACI.

Steps

1. Create LDIF files to capture the following ACI information.

   OID

      Step 1.3.6.1.4.1.30221.2.5.40

   Name

      Password Validation Details Requerst Control

Permission

read

The following examples show the example file contents for change password and password reset.

Example:`aci_toSvcAccount_forChangePassword.ldif`

```
# ACI to service account for change password
dn: uid=ssoDataStore,ou=ServiceAccounts,dc=example,dc=local
changetype: modify
add: aci
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.40")(version 3.0; acl "Access to the Password
Validation Details Request Control"; allow (read) userdn="ldap:///
uid=ssoDataStore,ou=ServiceAccounts,dc=example,dc=local";)
```

Example:`aci_toUsrAccount_forPasswordReset.ldif`

```
# ACI to a user account for password reset
dn: uid=user.7,ou=People,dc=example,dc=local
changetype: modify
add: aci
aci: (targetcontrol="1.3.6.1.4.1.30221.2.5.40")(version 3.0; acl "Access to the Password
Validation Details Request Control"; allow (read) userdn="ldap:///uid=user.
7,ou=People,,dc=example,dc=local";)
```

> **Note**
>
> For demonstration purposes, this sample LDIF file only targets one user. You can use other LDIF syntax to widen its coverage to include multiple users.

2. Use the `ldapmodify` command to configure the required ACI.

Example:

```
$ ldapmodify -f  <path>/aci_toSvcAccount_forChangePassword.ldif
-h  <host name>
-p  <LDAP port>
-D  <LDAP bind username>
-w  <LDAP bind password>
```

```
$ ldapmodify -f  <path>/aci_toUsrAccount_forPasswordReset.ldif
-h  <host name>
-p  <LDAP port>
-D  <LDAP bind username>
-w  <LDAP bind password>
```

> **Note**
>
> Line breaks are inserted for readability only.

Related links

- Managing Access Control in the PingDirectory Administration Guide

Defining a custom LDAP type for outbound provisioning

If you are using outbound provisioning and your directory server is not PingDirectory, Microsoft Active Directory, Oracle Unified Directory, or Oracle Directory Server, you can define a custom LDAP type for PingFederate to use to streamline the provisioning configuration.

Steps

1. Copy and rename `<pf_install>/pingfederate/server/default/conf/template/ldap-templates/sample.template.txt` file.
2. Change the `template.name` property value in the new template file.

   This property value appears in the LDAP Type list on the LDAP Configuration window when you save the template.
3. Modify other property values in the file to match the corresponding configuration of your directory server.

   These properties correspond to the fields shown on Outbound Provisioning > Channel > Source Settings. They help the provisioner determine when user records are added, changed, or removed.
4. Save the new template file.

   For a clustered PingFederate environment, perform these steps on the console node. No changes or restart of the PingFederate service is required on any nodes.

Next steps

After you have configured the LDAP type, you can create a new LDAP datastore using the newly defined LDAP type. To streamline outbound provisioning configuration, select the LDAP data store that uses the newly-defined LDAP type in the Source window.

Related links

- Identifying the source datastore
- Modifying source settings

Configuring other types of datastores

Besides connecting to a directory server using LDAP or a database server using Java Database Connectivity (JDBC), PingFederate can connect to other types of datastores, such as REST API-enabled data sources that return user attributes in JSON.

See the following topics for configuration steps:

- Configuring a REST API datastore
- Configuring a custom datastore
- Configuring an AWS DynamoDB datastore

Configuring a REST API datastore

To retrieve attribute data from a JSON-based REST API, you must first create a REST API datastore.

Steps

1. Go to System > Data & Credential Stores > Data Stores.
2. On the Data Stores window, click Add New Data Store.
3. On the Data Store Type tab, type a name for the datastore.
4. From the Type list, select Rest API.
5. Optional: To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log check box.
6. Click Next.
7. On the Configure Data Store Instance tab, click Add a new row to 'Base URLS and Tags'.
    1. Enter the Base URL of the data source offering REST API access to its data. You can enter multiple base URLs.

        > **Tip**
        >
        > If you configure multiple URLs, nodes use the first URL with a tag that matches one of the `node.tags` in the `run.properties` file.

    2. Optional: Enter one or more tags per base URL.

        Tags are defined in the `node.tags` property in the `<pf_install>/pingfederate/bin/run.properties` file. For a description of the `node.tags` property, see Deploying cluster servers.

        In PingFederate deployments that are regional, you can enter one or more tags for a Base URL, which specifies the PingFederate node the datastore should communicate with. If none of the tags match what is defined for the `node.tags` property, the default node is used.

    The following rules apply to tags:

    - You must separate multiple tags specified for one node with spaces.
    - Tags must unique per base datastore.
    - You cannot use a tag more than once per datastore.
    - Tags are optional. If needed, you can configure a non-default node without tags. Doing this is useful if you are not yet ready to tag the node, or if you are still in the planning stage but want to enter the address for the node now.

            1. Click Update under Action.
            2. Select Set as Default under Action beside the Base URL and Tags that you want to use as the default. The first Base URL and Tags configured is set as the default automatically.

> **Note**
>
> If the data source exposes multiple paths or requires specific query parameters to retrieve user records, enter the base URL here and then specify the path and query parameters in the attribute source configuration.
> For more information, see Specifying data source filters for REST API datastores.

8. If the data source requires specific HTTP request headers, click Add a new row to 'HTTP Request Headers'.

   > **Note**
   >
   > If configured, PingFederate includes the configured HTTP request headers and their values when contacting the data source.

   1. Enter the applicable name and value under Header Name and Header Value.
   2. Click Update under Action.

   Repeat these steps to define additional HTTP request headers and their values.

9. Click Add a new row to 'Attributes' to define local attribute names and map them to the data returned by the data source.

   Map each attribute to a path representing an attribute in the JSON response. This path follows the syntax defined in the JavaScript Object Notation (JSON) Pointer specification at tools.ietf.org/html/rfc6901.

   You must define at least one attribute.

   1. Enter the Local Attribute name and JSON Response Attribute Path.
   2. Click Update under Action.

      Repeat these steps to define additional attributes.

   > **Tip**
   >
   > Define only the attributes required by other configuration items, such as contract fulfillment or token authorization. Provide meaningful attribute names so that you can easily recognize them at a later time.

10. Select an Authentication Method:

Option	Description
None	PingFederate makes unauthenticated REST API requests to the data source. No credential information is required. This is the default setting.
Basic Authentication	PingFederate authenticates via the HTTP Basic authentication scheme. Enter the required credentials in the Username and Password fields.

Option	Description
OAuth 2.0 Bearer Token	PingFederate authenticates by presenting an OAuth 2.0 access token. In this scenario, PingFederate is an OAuth client, specifically a client that uses the client credential grant type to obtain access token from an authorization server and presents the access token to the data source for authentication. Enter the client credentials in the Client ID and Client Secret fields. Then enter the token endpoint URL at the authorization server and the applicable scope (or scopes) in the OAuth Token Endpoint and OAuth Scope fields.
Client TLS Certificate	To support mutual TLS (mTLS), PingFederate authenticates by presenting the client transport layer security (TLS) certificate that you select in the Client TLS Certificate list. If you have not yet created or imported a client certificate, click SSL Client Keys & Certificates to do so. For more information, see Manage SSL client keys and certificates.

11. Select the HTTP Method to call the REST API service: GET or POST.

    Both methods support the JSON and x-www-form-urlencoded body types, among others, but PingFederate can parse only JSON responses.

    > **Tip**
    >
    > When configuring data source filters for attribute sources and user lookup, you can define the body sent in GET or POST requests to the REST API data store.

12. Optional: To mask attribute values returned through this datastore in the PingFederate log, select the Mask Values in Log check box.

    This check box is visible only when editing an existing datastore and is not checked by default.

13. Optional: Click Show Advanced Fields to configure additional settings.

    The following table describes the advanced fields.

Field	Description
Enable HTTPS Hostname Verification	Indicates whether to verify that the hostname of the data source matches the subject (CN) or one of the subject alternative names (SANs) from the certificate. **Important** You should verify the hostname for all connections. This check box is selected by default.

Field	Description
Read Timeout (MS)	Defines the socket timeout in milliseconds. Enter `0` to set an infinite timeout. Enter a negative integer to use the default value set by the operating system. The default value is `10000` in milliseconds, which is 10 seconds.
Connection Timeout (MS)	Determines the timeout in milliseconds until a connection is established. Enter `0` to set an infinite timeout. Enter `-1` to use the default value set by the operating system. The default value is `10000` in milliseconds, which is 10 seconds.
Max Payload Size (KB)	Defines the maximum allowed size in kilobytes (KB) of the returned JSON response payload. Enter `0` to configure an unrestricted payload size. The default value is `1024` in KB.
Retry Request	Determines whether to retry a user data retrieval request if the data source returns an HTTP status code found in the Retry Error Codes. This check box is selected by default.
Maximum Retries Limit	Defines the maximum number of retry attempts if the data source returns an HTTP status code found in the Retry Error Codes. The default value is `5`.
Retry Error Codes	Enter a comma-separated list of HTTP status codes, for which if received from the data source, PingFederate might retry the request. For example, you can enter 429 for "Too Many Request" or 503 for "Service Unavailable". The default value is `429`.
Test Connection URL	Determines the URL to which PingFederate sends requests to test the datastore connection on the Actions tab. When not specified (the default), PingFederate sends requests to the base URL of the datastore.

14. On the Actions tab, verify the datastore configuration.

    1. Click Test Connection to test the connectivity between PingFederate and the data source.

       Result:

       The administrative console displays the results returned by the data source. The PingFederate server log may contain additional messages as well.

    2. Review the results.

    3. Optional: Click Reset and repeat the test.

15. On the Summary tab, review your configuration, amend as needed, click Save to keep your configuration or click Cancel to discard it.

Example

You have two use cases that can leverage user attributes obtained through REST APIs. The data source returns user records in JSON:

```
{
  "uid": "asmith",
  "office": {
    "city": "Denver",
    "state": "CO",
    "zipCode": 80202
  },
  "telephoneNumbers": [
    "+1 303-555-1234",
    "+1 303-555-5678"
  ],
  "department": "Engineering"
}
```

The first use case requires the user's department, while the second use case requires the first telephone number and the ZIP code.

To address both use cases, create a REST API datastore with the following attributes.

Local Attribute	JSON Response Attribute Path
Dept	/department
Telephone	/telephoneNumbers/0
Zip	/office/zipCode

Once set up, you can fulfill various contracts or configure issuance criteria based on the attribute data from the data source.

Related links

- Token authorization
- Fulfillment by datastore queries

Configuring a custom datastore

You can configure your own custom datastore instance to perform specified actions.

About this task

Developers can use the PingFederate SDK to create specific drivers for non-Java Database Connectivity (JDBC) , LDAP datastores, or more sophisticated JDBC or LDAP queries, including flat files or SOAP-connected databases. You can write datastores to perform configuration assistance or validation actions, such as testing a connection to a database. Actions can also include generation of parameters that might need manual setting in a configuration file.

For more information, see the Javadoc for the `CustomDataSourceDriver` interface, the `SamplePropertiesDataStore.java` file for a sample implementation, and the SDK Developer’s Guide for build and deployment information.

> **Tip**
>
> The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

Steps

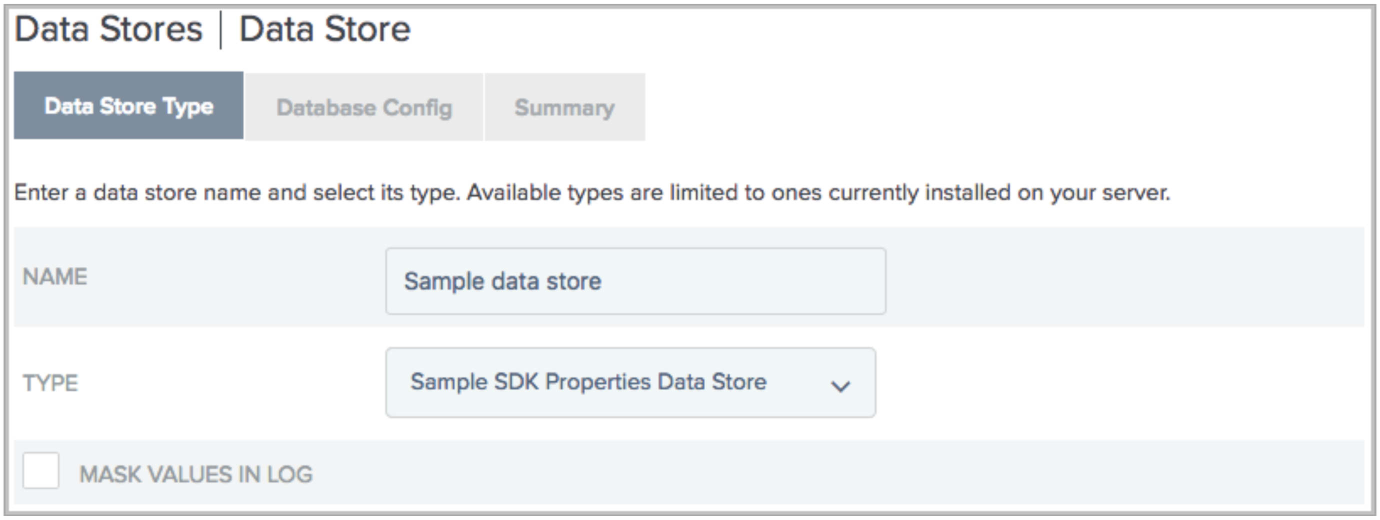
1. After the data store driver (JAR) file is written and installed, select it in the Data Store window when creating a new instance of your data store.

2. On the Configure Data Store Instance tab, configure your data store instance.

   > **Note**
   >
   > Depending on the data store implementation, configuration requirements vary.

   Example:

   After building and deploying the sample from the `<pf_install>/pingfederate/sdk/plugin-src/custom-data-store-example` directory, you can create an instance of the Sample SDK Properties Data Store and configure the rest, as shown in the following images.

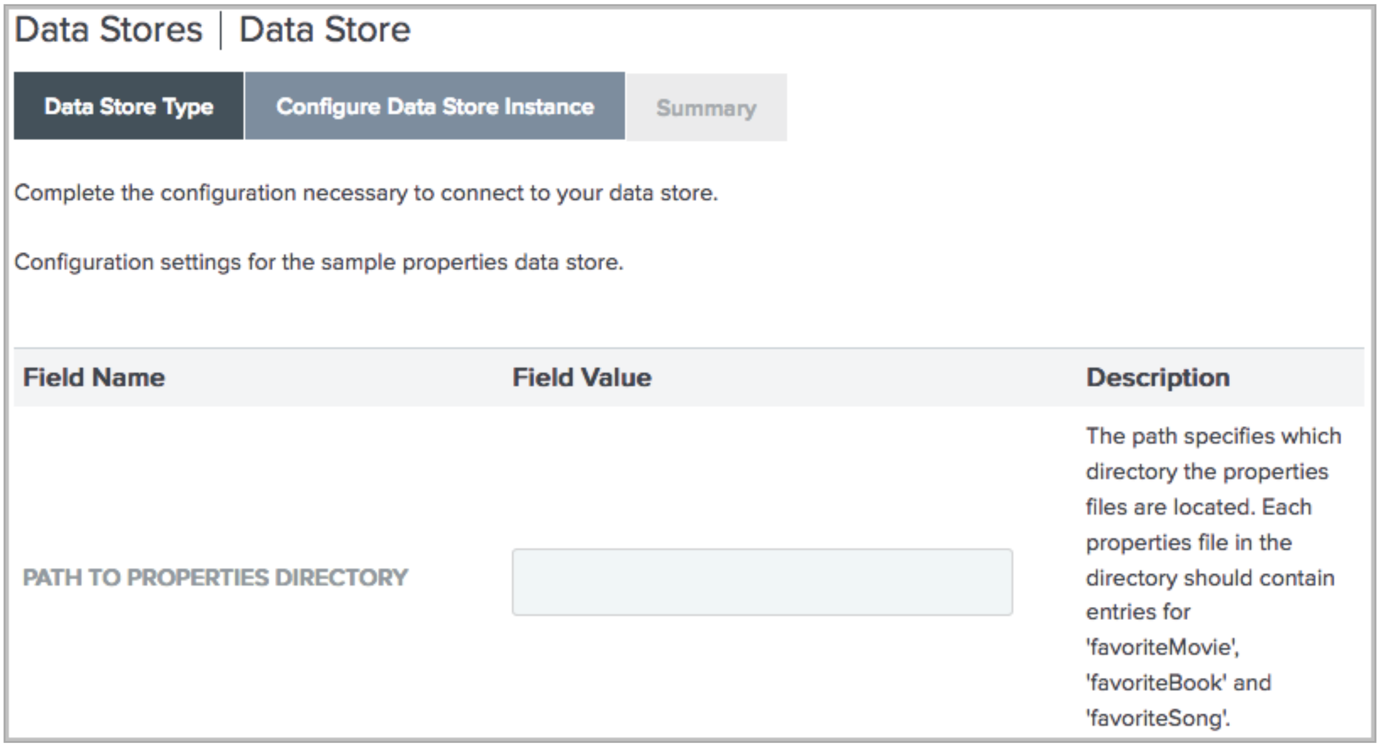


Enter a data store name and select its type. Available types are limited to ones currently installed on your server.

NAME: Sample data store

TYPE: Sample SDK Properties Data Store

□ MASK VALUES IN LOG



Data Stores | Data Store

Complete the configuration necessary to connect to your data store.

Configuration settings for the sample properties data store.

Field Name	Field Value	Description
PATH TO PROPERTIES DIRECTORY		The path specifies which directory the properties files are located. Each properties file in the directory should contain entries for 'favoriteMovie', 'favoriteBook' and 'favoriteSong'.

> **Note**
>
> When editing an existing instance, you can modify the name of the data store instance and toggle the option for if PingFederate should mask attribute values returned from this data store instance in PingFederate logs.

3. On the Actions window, follow the on-screen instructions provided by the developer to complete any required tasks.

> **Note**
>
> Depending on the datastore implementation, configuration requirements vary. If no action is required, this window is not shown.

4. Click Save to keep your configuration.

Configuring an AWS DynamoDB datastore

Set up an Amazon Web Services (AWS) DynamoDB so that PingFederate can store user attributes in the DynamoDB NoSQL database.

Before you begin

Ensure that your server is configured to access DynamoDB. For more information on how to configure your server to access DynamoDB, see Setting up DynamoDB (web service) in the AWS DynamoDB documentation.

About this task

DynamoDB's NoSQL nature allows for flexible schema design and horizontal scalability, accommodating varying attribute types and high volumes of user data. DynamoDB's robust security and reliability features help ensure the confidentiality and integrity of stored user attributes.

To create a DynamoDB datasource and map local attribute names to DynamoDB document paths:

Steps

1. Go to System > Data & Credential Stores > Data Stores.
2. In the Data Stores window, click Add New Data Store.
3. On the Data Store Type tab, enter a name for the datastore.
4. Optional: To mask attribute values returned from this datastore in PingFederate logs, select the Mask Values in Log check box.
5. In the Type list, select AWS DynamoDB.
6. Click Next.
7. In the Configure Data Store Instance window, configure your AWS DynamoDB connection.
8. In the Attributes field, define the list of attributes that you want the datastore to return when performing a lookup.

   For information about each field, see the following table.

Field	Description
Local Attribute	The attribute names that are populated in drop-down menus during contract mapping.
DynamoDB Attributes	Specifies document path, the DynamoDB-specific syntax that identifies where precisely in the record an attribute is located. For more information, see .aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.Attributes.html//[Document paths^] in the AWS DynamoDB documentation.
Table Name	The name of the DynamoDB table.
Allow Multi-value Attributes	When selected, a DynamoDB query that returns multiple records will result in multi-valued attributes. Otherwise, only the first record returned from the query is used. This check box is selected by default.
API Call Timeout	The amount of time in milliseconds to allow the client to complete the execution of the API call. The default value is 10000.
API Call Attempt Timeout	The amount of time in milliseconds to wait for the HTTP request to complete before giving up and timing out. The default value is 1000.

Field	Description
Mask Values in Log	Determines whether all attribute values returned through this datastore should be masked in PingFederate logs. These values are only applicable when editing an existing data store.

9. Click Next.
10. Click Test Connection to determine whether the administrative node can query the specified DynamoDB table.

> **Note**
>
> Datastore validation is not enabled during configuration, which lets you configure datastores without requiring a successful connection between the administrative node and the AWS DynamoDB. You can also save the datastore even if the connection is not currently successful.

Next steps

See Specifying filters and fields for an AWS DynamoDB datastore to continue setting up your DynamoDB datasource and map local attribute names to DynamoDB document paths.

Defining a datastore for persistent authentication sessions

When enabling PingFederate authentication sessions, you can select the persistent option so that PingFederate can leverage previous sessions as users request protected resources after restarting their browsers.

About this task

This optional persistent configuration requires external storage of session-state data, as opposed to in-memory alone. By default, PingFederate uses its internal HSQLDB database to maintain persistent authentications. You can configure PingFederate to maintain persistent authentication sessions externally on a database server or a PingDirectory server. Also, the PingFederate SDK lets you use custom solutions for persistent session storage.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Steps

1. Create the required data structure on the external storage medium.
2. Modify two PingFederate configuration XML files.

Related links

- Sessions
- Server Clustering Guide

Configuring an external database for authentication sessions

Set up various tables so that PingFederate can store authentication sessions on corresponding database servers.

About this task

Specific tables are required in order for PingFederate to store authentication sessions on your database server. Table-setup scripts are provided for supported database servers.

Steps

1. Run the table-setup scripts, provided in the `<pf_install>/pingfederate/server/default/conf/authentication-session/sql-scripts` directory, for your database server.
2. If you have not already done so, go to System > Data & Credential Stores. In the Data Stores window, create a Java Database Connection (JDBC) datastore for your database server.
3. Copy the system ID of the applicable JDBC datastore from the Data Stores window.
4. Edit the `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   > **Note**
   >
   > For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System** > **Server** > **Cluster Management** as explained in later steps.

   Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

   Example:

   For example, if the system ID is `JDBC-123456789ABCDEF123456789ABCDEF123456A0A6`, update the `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl.xml` file as follows.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <c:config xmlns:c="http://www.sourceid.org/2004/05/config">
       <c:item name="PingFederateDSJNDIName">JDBC-123456789ABCDEF123456789ABCDEF123456A0A6</c:item>
   </c:config>
   ```
5. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   For example, if the system ID is `SessionStorageManager`:

   1. Go to the `# Service for storing Authentication Sessions` section.

```
# Service for storing Authentication Sessions.
# Supported classes:
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl : Use this
service-point for a Jdbc implementation.
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl : Use this
service-point for an LDAP implementation.
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl : Use
this service-point for a DynamoDB implementation.
session.storage.manager=org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
```

    2. Validate that the value of the `session.storage.manager` service is `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl`, the default value.

    > **Note**
    >
    > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

6. Start or restart the PingFederate service.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

Result

PingFederate removes expired authentication sessions from the database once a day. To fine-tune the frequency and the number of expired authentication sessions to remove, see Managing authentication sessions stored in the database.

Related links

- Configuring a JDBC connection
- System requirements

Configuring PingDirectory for authentication sessions

Use specific schema objects to enable PingFederate to store authentication sessions on your directory server. For PingDirectory, LDIF scripts are provided for this purpose.

Steps

1. Update the LDAP schema.
    1. Sign on to the PingDirectory administrative console.
    2. Go to LDAP Schema > Schema Utilities.
    3. Click Import Schema Element.
    4. Copy the schema changes from the `authentication-session-attributes-ldap-pingdirectory.ldif` file and paste them into the text area.

    The file is located in the `<pf_install>/pingfederate/server/default/conf/authentication-session/ldif-scripts` directory.

    Replace the placeholder values with relevant information from your directory server.

  5. Click Import.

2. Create the following indexes.

Attribute name	Index type
`pf-authn-session-group-hashed-session-id`	equality
`pf-authn-session-group-user-ids`	equality
`pf-authn-session-group-expiry-time`	ordering
`pf-authn-session-group-last-activity-time`	ordering

Create these indexes with PingDirectory's **`dsconfig`** utility. The **`dsconfig`** utility is interactive. You can also provide inputs as command arguments. The following examples create the indexes.

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name pf-authn-session-group-hashed-session-id \
  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name pf-authn-session-group-user-ids \
  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name pf-authn-session-group-expiry-time \
  --set index-type:ordering
```

```
$ bin/dsconfig create-local-db-index \
  --backend-name userRoot \
  --index-name pf-authn-session-group-last-activity-time \
  --set index-type:ordering
```

After adding the indexes, use the **`rebuild-index`** utility to build the indexes. The following example builds the required indexes. x

```
$ bin/rebuild-index \
  --baseDN "dc=example,dc=com" \
  --index pf-authn-session-group-hashed-session-id \
  --index pf-authn-session-group-user-ids \
  --index pf-authn-session-group-expiry-time \
  --index pf-authn-session-group-last-activity-time
```

For more information, see Working with Indexes in the PingDirectory Administration Guide.

3. If you have not already done so, create an LDAP data store for your directory server on System > Data & Credential Stores > Data Stores.

4. Copy the system ID of the applicable LDAP data store from the Data Stores window.

5. Edit the `/pingfederate/server/default/data/config-store/org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl.xml` file.

> **Note**
>
> For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System** > **Server** > **Cluster Management** as explained in later steps.

   1. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

      Example:

      For example, if the system ID is `LDAP-123456789ABCDEF123456789ABCDEF123456A0AC`, update the configuration file as follows.

      ```
      ...
      <!-- Data store id -->
      <c:item name="PingFederateDSJNDIName">LDAP-123456789ABCDEF123456789ABCDEF123456A0AC</c:item>
      ...
      ```

   2. Enter a value for the `<c:item name="SearchBase"/>` element.

      > **Tip**
      >
      > This is the distinguished name (DN) that points to the client location. For more information, see the inline comment and the LDIF scripts in the `<pf_install>/pingfederate/server/default/conf/authentication-session/ldif-scripts` directory.

   3. Update the attribute names only if you have changed attribute names in the LDIF scripts located in the `<pf_install>/pingfederate/server/default/conf/authentication-session/ldif-scripts` directory.

   4. Save the file.

6. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storing Authentication Sessions` section.

```
# Service for storing Authentication Sessions.
# Supported classes:
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl : Use this
service-point for a Jdbc implementation.
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl : Use this
service-point for an LDAP implementation.
#    org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl : Use
this service-point for a DynamoDB implementation.
session.storage.manager=org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
```

2. Change the value of the `session.storage.manager` service to `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl`.

> **Note**
>
> For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can't replicate this change to other nodes.

7. Start or restart the PingFederate service.

> **Note**
>
> PingFederate was tested with vendor-specific JDBC 4.2 drivers. For more information, see Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

> **Note**
>
> When storing persistent authentication sessions on a PingDirectory server, you must also configure a cleanup plugin in PingDirectory to remove expired authentication sessions from your directory server. For more information, see Managing authentication sessions stored in PingDirectory.

Related links

- System requirements
- Installing the Server from the PingDirectory Administration Guide
- Using the Schema Editor Utilities from the PingDirectory Administration Guide
- Working with Indexes from the PingDirectory Administration Guide
- Configuring an LDAP connection

Configuring an AWS DynamoDB for persistent authentication sessions

Set up an Amazon Web Services (AWS) DynamoDB so that PingFederate can store persistent authentication sessions in the DynamoDB NoSQL database.

Before you begin

Ensure that your server is configured to access DynamoDB.

About this task

PingFederate requires specific tables to store persistent authentication sessions on your DynamoDB server. Table-setup scripts are provided for this purpose.

> **Note**
>
> PingFederate supports the use of global multi-region tables for DynamoDB. However, these tables are managed entirely by AWS.
> Learn more about configuring global tables in Amazon DynamoDB global tables in the AWS documentation.

Steps

1. **To create a table in DynamoDB to contain authentication sessions, run the commands in the `<pf_install>/pingfederate/server/default/conf/authentication-session/nosql-scripts/authentication-session-dynamodb.txt` file.**

   This file contains basic commands to create the table, with sample values for read and write throughput, as well as the command to enable `ExpiryTime` as the Time-to-Live (TTL) attribute.

   1. **Optional: To rename the table and index names, edit the `table-name` and `\"IndexName\"` values in the table script in the `authentication-session-dynamodb.txt` file.**

2. **Optional: If authentication sessions are not already enabled in PingFederate, go to Authentication > Policies > Sessions to configure them. For more information, see Configuring authentication sessions.**

3. **Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file:**

   1. **Locate the `SessionStorageManager` service point:**

   ```
   # Service for storing Authentication Sessions.
   # Supported classes:
   # org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl : Use this service-
   point for a Jdbc implementation.
   # org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl : Use this service-
   point for an LDAP implementation.
   # org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl : Use this
   service-point for a DynamoDB implementation.
   session.storage.manager=org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
   ```

   1. **Update the value of the service point to `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl`.**

   2. **Save the file.**

   > **Note**
   >
   > For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can’t replicate this change to other nodes.

4. **Optional: If you modified the default table and index names in the `authentication-session-dynamodb.txt` file in step 1, edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl.xml` file to match your customized configuration.**

Note

If you ran the script commands from the `authentication-session-dynamodb.txt` as is and did not change the default names in the commands, you do not need to edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl.xml` file.

1. Replace the `<c:item name="SessionGroupTableName"/>`, `<c:item name="UserIdTableName"/>`, `<c:item name="HashedSessionIdIndexName"/>`, `<c:item name="SessionUserIdGroupIdIndexName"/>` element values with the customized names created during your initial DynamoDB setup.

2. Save the file.

   The following table describes the preconfigured PingFederate variables in the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl.xml` file.

DynamoDB Session Storage Manager Variables

Variable	Description
`PingFederateAuthenticationSessionHashedSessionId-index`	The name of the hashed session ID index. This is the default value.
`PingFederateAuthenticationSessionUserIdGroupId-index`	The name of the user ID and group ID index. This is the default value.
`EndpointOverride`	An optional endpoint URL which should not be used in production but allows for testing with a local development DynamoDB instance. By default, this value is empty. To test DynamoDB running locally, specify `EndpointOverride` to point to a local endpoint. For example, `<c:item name="EndpointOverride">http://localhost:8000</c:item>`. For more information, see DynamoDB local usage notes in the AWS DynamoDB documentation.
`dynamoDbBatchSize`	Number of records to request when performing batch operations against DynamoDB. The minimum allowed value is one, the maximum allowed value is 100, and the default value is 50.
`ApiCallTimeout`	The amount of time in milliseconds to allow the client to complete the execution of the API call. The default value is 10000.
`ApiCallAttemptTimeout`	The amount of time in milliseconds to wait for the HTTP request to complete before giving up and timing out. The default value is 1000.

5. Start or restart the PingFederate service.

> **Note**
>
> For a clustered PingFederate, replicate this new configuration to other engine nodes on **System** > **Server** > **Cluster Management**. Start or restart the PingFederate service on each engine node to active the change.

Result

PingFederate relies on the DynamoDB TTL attribute to remove expired authentication sessions from the database. For more information on TTL, see Expiring items by using DynamoDB Time to Live (TTL) in the AWS DynamoDB documentation.

Using custom solutions for persistent session storage

The PingFederate SDK supports custom storage for persistent authentication sessions.

Steps

1. Implement the `SessionStorageManager` interface.

   > **Note**
   >
   > For more information, see the Javadoc for the `SessionStorageManager` interface. The Javadocs for PingFederate are in the `<pf_install>/pingfederate/sdk` directory.

2. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file:

   1. Go to the `# Service for storing Authentication Sessions` section.

      ```
      # Service for storing Authentication Sessions.
      # Supported classes:
      #   org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl : Use this
      service-point for a Jdbc implementation.
      #   org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl : Use this
      service-point for an LDAP implementation.
      #   org.sourceid.saml20.service.session.data.impl.SessionStorageManagerDynamoDBImpl : Use this
      service-point for a DynamoDB implementation.
      session.storage.manager=org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
      ```

   2. Change the value of the `session.storage.manager` service to the name of your class.

      > **Note**
      >
      > For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can't replicate this change to other nodes.

3. Deploy the required program files of your custom implementation to all PingFederate servers.

4. Start or restart PingFederate.

> **Note**
>
> PingFederate was tested with vendor-specific JDBC 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

OAuth grant datastores

Learn about persistent grant data stores and persistent authorizations.

PingFederate uses a built-in HSQLDB as its persistent grant datastore after the initial setup.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it may cause various problems due to its limitations and HSQLDB involved cases are not supported by PingIdentity.

Persistent authorizations include those obtained by OAuth clients in the following ways:

- Grants obtained or updated using the authorization code, resource owner credentials, or device authorization grant type, in conjunction with the refresh token grant type

> **Note**
>
> If the use cases involve mapping attributes from authentication sources, such as IdP adapter instances or IdP connections, or password credential validator (PCV) instances to the access tokens, directly or through persistent grant-extended attributes, storing these attributes from authentication sources and their values along with the persistent grants maintains them for reuse when clients subsequently present refresh tokens for new access tokens.

- Grants obtained or updated by using the implicit grant type, for which PingFederate is configured to reuse existing persistent grants

> **Note**
>
> If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens, runtime procedures obtain attribute values for each token request, but persistent grants do not store with attributes or their values.

Persistent grants and any associated attributes and their values remain valid until the grants expire or until PingFederate explicitly revokes or cleans them up.

> **Note**
>
> Attribute values are always stored encrypted when a directory is used. If a database server is used (including the built-in HSQLDB database), attribute values are also stored encrypted by default.

Related links

- OAuth persistent grants cleanup

- Grant types
- Server Clustering Guide

Configuring external databases for grant storage

Specific tables are required in order for PingFederate to store grants, the associated attributes, and their values (if any), on your database server. Table-setup scripts are provided for supported database servers.

About this task

Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file and the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

Steps

1. Run the table-setup scripts for your database server provided in the `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts` directory.

2. If you have not already done so, create a JDBC data store for your database server on System > Data & Credential Stores > Data Stores.

3. Copy the System ID of the applicable Java Database Connection (JDBC) data store from the Data Stores window.

4. Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file.

   > **Note**
   >
   > For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System > Server > Cluster Management** as explained in later steps.

   1. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

      Example:

      If the system ID is `JDBC-123456789ABCDEF123456789ABCDEF123456A0A6`, update the `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<c:config xmlns:c="http://www.sourceid.org/2004/05/config">
    <c:item name="PingFederateDSJNDIName">JDBC-123456789ABCDEF123456789ABCDEF123456A0A6</c:item>
</c:config>
```

5. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storage of access grants` section.

   2. Change the `access.grant.manager` service endpoint to the following:

```
...
access.grant.manager=org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl
...
```

> **Note**
>
> In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

6. Start or restart PingFederate.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

Result

PingFederate provides two cleanup tasks for persistent grants. One task manages expired grants, while another task caps the number of grants based on a combination of user, client, grant type, and authentication context.For more information, see OAuth persistent grants cleanup.

Related links

- Configuring a JDBC connection
- System requirements

Configuring directories for grant storage

PingFederate requires specific schema objects to store grants, the associated attributes, and their values (if any) on your directory server. LDIF scripts are provided for supported directory servers.

Steps

1. Review the LDIF scripts for your directory server provided in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.
2. Replace placeholder values with relevant information from your directory server.
3. Run the LDIF scripts to update your LDAP schema.

> **Note**
>
> For Active Directory, run the script to create the attributes, then run the script to create the object class.
> For PingDirectory, run the **`ldapmodify`** command. For example:
> `ldapmodify --defaultAdd --filename "<path to ldif file>\access-grant-ldap-pingdirectory.ldif" -h <hostname> -p <port> -D "<adminDN>" -w <adminPassword>`

4. If you have not already done so, create an LDAP datastore for your directory server on System > Data & Credential Stores > Data Stores.

5. Copy the system ID of the applicable LDAP datastore from **System** > **Data & Credential Stores** > **Data Stores**.

6. Edit the configuration file relevant to your directory server.

> **Note**
>
> This configuration file is located in the `<pf_install>/pingfederate/server/default/data/config-store` directory, as described in the following table.

Directory server	Configuration file
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl.xml`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl.xml`
Oracle Unified Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl.xml`

> **Note**
>
> For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System** > **Server** > **Cluster Management** as explained in later steps.

   1. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your datastore connection.

      Example:

      If the system ID is `LDAP-123456789ABCDEF123456789ABCDEF123456A0A6`, update the configuration file as follows.

```
...
<!-- Data store id -->
<c:item name="PingFederateDSJNDIName">LDAP-123456789ABCDEF123456789ABCDEF123456A0A6</c:item>
...
```

   2. Enter a value for the `<c:item name="SearchBase"/>` element.

      > **Tip**
      >
      > This is the distinguished name (DN) that points to the access grants location. Learn more in the inline comment and LDIF scripts in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.

   3. Update the attribute names only if you have changed attribute names in the LDIF scripts located in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.

   4. Save the file.

7. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storage of access grants` section.

```
# Service for storage of access grants
# Supported classes:
#    org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl : Use this service-point for a Jdbc
implementation
#    org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl : Use this service-point for a
Microsoft Active Directory implementation
#    org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl : Use this service-point for
an Oracle Directory Server Enterprise Edition implementation
#    org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl : Use this service-
point for a PingDirectory implementation
#    org.sourceid.oauth20.token.AccessGrantManagerDynamoDBImpl : Use this service-point for an
Amazon DynamoDB implementation
access.grant.manager=org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl
```

2. Change the value of the `access.grant.manager` service to one of the following values.

Directory server	Service value
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl`
Oracle Unified Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl`

> **Note**
>
> For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can't replicate this change to other nodes.

8. Start or restart PingFederate.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

9. In the directory, create indexes for the following OAuth grant attributes. If you are using Active Directory, skip this step because the script for Active Directory creates these indexes. If you are using PingDirectory, learn more in Indexing grant attributes in PingDirectory for more information.

Attribute name	Index type
`accessGrantGuid`	equality
`accessGrantUniqueUserIdentifier`	equality
`accessGrantHashedRefreshTokenValue`	equality

Attribute name	Index type
`accessGrantClientId`	equality
`accessGrantExpires`	ordering

Result

PingFederate provides two cleanup tasks for persistent grants. One task manages expired grants, while another task caps the number of grants based on a combination of user, client, grant type, and authentication context. Learn more in OAuth persistent grants cleanup.

Related links

- Configuring an LDAP connection
- System requirements

Indexing grant attributes in PingDirectory

If you use PingDirectory, or another directory, to store OAuth persistent grants for PingFederate, you must index the grant attributes.

About this task

Index these OAuth grant attributes using the procedure below.

Attribute name	Index type
`accessGrantGuid`	equality
`accessGrantUniqueUserIdentifier`	equality
`accessGrantHashedRefreshTokenValue`	equality
`accessGrantClientId`	equality
`accessGrantExpires`	ordering

Steps

1. Create the indexes using the PingDirectory `dsconfig` utility.

   The `dsconfig` utility is interactive, letting you enter command arguments. The following examples create the required indexes.

```
$ bin/dsconfig create-local-db-index \
                  --backend-name userRoot \
                  --index-name accessGrantGuid \
                  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
                  --backend-name userRoot \
                  --index-name accessGrantUniqueUserIdentifier \
                  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
                  --backend-name userRoot \
                  --index-name accessGrantHashedRefreshTokenValue \
                  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
                  --backend-name userRoot \
                  --index-name accessGrantClientId \
                  --set index-type:equality
```

```
$ bin/dsconfig create-local-db-index \
                  --backend-name userRoot \
                  --index-name accessGrantExpires \
                  --set index-type:ordering
```

2. After adding the indexes, build the indexes using the `rebuild-index` utility.

   The following example builds the required indexes.

```
$ bin/rebuild-index \
                  --baseDN "dc=example,dc=com" \
                  --index accessGrantGuid \
                  --index accessGrantUniqueUserIdentifier \
                  --index accessGrantHashedRefreshTokenValue \
                  --index accessGrantClientId \
                  --index accessGrantExpires
```

> **Note**
>
> You can configure a PingDirectory plugin to handle the cleanup of expired persistent grants and the associated attributes. The plugin allows fine-grained control over various aspects of the cleanup task, which can smooth out the performance impact. For more information, see Managing expired persistent grants in PingDirectory.

Related links

- PingDirectory: Working with indexes

Using custom solutions for grant storage

Use the PingFederate SDK to implement a custom solution for grant storage.

Steps

1. Implement the `AccessGrantManager` interface.

   > **Note**
   >
   > For more information, see the Javadoc for the `AccessGrantManager` interface, the `SampleAccessGrant.java` file for a sample implementation, and the SDK Developer's Guide for build and deployment information.

   > **Tip**
   >
   > The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

2. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storage of access grants` section.

      ```
      # Service for storage of access grants
      # Supported classes:
      #    org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl : Use this service-point for a Jdbc implementation
      #    org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl : Use this service-point for a Microsoft Active Directory implementation
      #    org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl : Use this service-point for an Oracle Directory Server Enterprise Edition implementation
      #    org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirectoryImpl : Use this service-point for a PingDirectory implementation
      #    org.sourceid.oauth20.token.AccessGrantManagerDynamoDBImpl : Use this service-point for an Amazon DynamoDB implementation
      access.grant.manager=org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl
      ```

   2. Change the value of the `access.grant.manager` service to the name of your class.

   > **Note**
   >
   > For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can't replicate this change to other nodes.

3. Deploy the required program files of your custom implementation to all PingFederate servers.

4. Start or restart PingFederate.

   > **Note**
   >
   > PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

Configuring an Amazon Dynamo database for persistent grants

About this task

Maintain access grants in Amazon Web Services (AWS) native DynamoDB.

Global secondary indexes:

- "AccessGrantsUniqueUserId-index" Hash: UniqueUserID
- "AccessGrantsHashedRefreshToken-index" Hash: HashedRefreshToken
- "AccessGrantsClientId-index": Hash ClientID
- "AccessGrantsLimitCheck-index" Hash: UniqueUserID, Range: ClientID

Steps

1. To create a table in the DynamoDB to contain access grants, run the commands located in the `access-grant-dynamodb.txt` file in the `server/default/conf/access-grant/nosql-scripts` directory.

   This file contains the create table command, the key, global secondary indexes, and the attributes needed for the key and index, as well as the command to enable the TTL expires attribute.

2. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storage of access grants` section.
   2. Change the `access.grant.manager` service endpoint to the following:

      ```
      ...
      access.grant.manager=org.sourceid.oauth20.token.AccessGrantManagerDynamoDBImpl
      ...
      ```

      > **Note**
      >
      > In clustered PingFederate environments, you must manually edit the `service-points.conf` file on each node because cluster replication can't replicate this change to other nodes.

3. Edit the `server/default/data/config-store/org.sourceid.oauth20.token.AccessGrantManagerDynamoDBImpl.xml` file.

PingFederate Access Grants

Access Grants	Description
`AccessGrantsUniqueUserId-index`	The name of the user ID index. This is the default value.
`AccessGrantsHashedRefreshToken-index`	The name of the hashed refresh token index. This is the default value.

Access Grants	Description
`AccessGrantsClientId-index`	The name of the client ID index. This is the default value.
`AccessGrantsLimitCheck-index`	The name of the limits check index. This is the default value.
`EndpointOverride`	An optional endpoint URL which should not be used in production but allows for testing with a local development DynamoDB instance. By default, this value is empty. To test DynamoDB running locally, specify `EndpointOverride` to point to a local endpoint. For example, `<c:item name="EndpointOverride">http://localhost:8000</c:item&gt`; . For more information, see DynamoDB local usage notes in the AWS DynamoDB documentation.

4. Export the AWS region.

   See the following for a sample command `export AWS_REGION=us-east-2`.

5. Start or restart PingFederate.

OAuth client datastores

Change the default storage method of XML files in PingFederate to make it easier to register clients or manage their records through the OAuth Client Management Service.

PingFederate stores client records in XML files by default. On-disk storage allows you to manage clients using the administrative console and the administrative API. Client records are part of the configuration archive.

You can configure PingFederate to store client records externally, which provides the flexibility to manage client records through the OAuth Client Management Service, or enable dynamic client registration for your partner-developers. In this scenario, client records are not part of the configuration archive. They are stored on a database server, a directory server, or some other storage medium through the use of the PingFederate SDK

Related links

- Managing OAuth clients
- Configuration archive
- PingFederate administrative API
- OAuth Client Management Service
- Dynamic client registration
- Server Clustering Guide

Configuring external databases for client storage

Specific tables are required in order for PingFederate to store OAuth client records on your database server. Table-setup scripts are provided for supported database servers.

About this task

> **Caution**
>
> PingFederate does not migrate client records from one storage medium to another. You must recreate your clients after updating the client storage configuration. If you need only a few clients, you can recreate them using the administrative console.
> If you need a large number of clients, use the administrative API to retrieve your client records before updating the client storage. Update the client storage configuration and recreate your clients using the administrative API based on the retrieved records. For more information, see PingFederate administrative API.

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.
    1. Locate the `client.manager` service point.

```
# Service for storing OAuth client configuration.
# Supported classes are
#   org.sourceid.oauth20.domain.ClientManagerXmlFileImpl  : Use this service-point for an XML
implementation.
#   org.sourceid.oauth20.domain.ClientManagerJdbcImpl     : Use this service-point for a Jdbc
implementation.
#   org.sourceid.oauth20.domain.ClientManagerLdapImpl     : Use this service-point for an LDAP
implementation.
#   org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl : Use this service-point for a
DynamoDB implementation
#   org.sourceid.oauth20.domain.ClientManagerGenericImpl  : Use this service-point if you have
specified a custom ClientStorageManager implementation above.
client.manager=org.sourceid.oauth20.domain.ClientManagerXmlFileImpl
```

    2. Update the value of the service point to `org.sourceid.oauth20.domain.ClientManagerJdbcImpl`.
    3. Save the file.

> **Important**
>
> You must set up an external database because you cannot share the bundled HSQLDB database across multiple PingFederate engine nodes. For production standalone deployments, we recommend you store the client records in an external secured database.

2. Run the table-setup scripts for your database server provided in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/sql-scripts` directory.

3. If you have not already done so, create a Java Database Connectivity (JDBC) datastore for your database server. Go to = System > Data & Credential Stores.

4. In the Data Stores window, copy the system ID of the applicable JDBC datastore.

5. Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.domain.ClientManagerJdbcImpl.xml` file.

   > **Note**
   >
   > For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System** > **Server** > **Cluster Management** as explained in later steps.

   Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your datastore connection and save the file.

   Example:

   If the system ID is `JDBC-123456789ABCDEF123456789ABCDEF123456A0AC`, update the `org.sourceid.oauth20.domain.ClientManagerJdbcImpl.xml` file as follows.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <c:config xmlns:c="http://www.sourceid.org/2004/05/config">
       <c:item name="PingFederateDSJNDIName">JDBC-123456789ABCDEF123456789ABCDEF123456A0AC</c:item>
   </c:config>
   ```

6. Start or restart PingFederate.

   > **Note**
   >
   > PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

Related links

- Configuring a JDBC connection
- System requirements

Configuring an Amazon DynamoDB for client storage

Set up an Amazon DynamoDB so that PingFederate can store OAuth client records in the DynamoDB NoSQL database.

Before you begin

Ensure that your server is configured to access DynamoDB.

About this task

PingFederate requires specific tables to store OAuth client records on your DynamoDB server. Table-setup scripts are provided for this purpose.

Steps

1. To create a table in DynamoDB to contain OAuth clients, run the commands in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/nosql-scripts/oauth-client-management-dynamodb.txt` file.
    1. Optional: To rename the table and index names, edit the `table-name` and `\"IndexName\"` values in the table script in the `oauth-client-management-dynamodb.txt` file.
2. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file:
    1. Locate the `ClientManager` service point:

    ```
    # Service for storing OAuth client configuration.
    # Supported classes are
    # org.sourceid.oauth20.domain.ClientManagerXmlFileImpl : Use this service-point for an XML
    implementation.
    # org.sourceid.oauth20.domain.ClientManagerJdbcImpl : Use this service-point for a Jdbc
    implementation.
    # org.sourceid.oauth20.domain.ClientManagerLdapImpl : Use this service-point for an LDAP
    implementation.
    # org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl : Use this service-point for a
    DynamoDB implementation
    # org.sourceid.oauth20.domain.ClientManagerGenericImpl : Use this service-point if you have
    specified a custom ClientStorageManager implementation above.
    client.manager=org.sourceid.oauth20.domain.ClientManagerXmlFileImpl
    ```

    2. Update the value of the `class` attribute to `org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl`.
    3. Save the file.

    > **Note**
    >
    > For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can't replicate this change to other nodes.

3. Optional: If you modified the default table and index names in the `oauth-client-management-dynamodb.txt` file in step 1, edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl.xml` file to match your customized configuration.

    > **Note**
    >
    > If you ran the script commands from the `oauth-client-management-dynamodb.txt` as is and did not change the default names in the commands, you do not need to edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl.xml` file.

    1. Replace the `<c:item name="ClientsTableName"/>`, `<c:item name="ClientIdIndex"/>`, `<c:item name="ClientNameIndex"/>`, `<c:item name="LastModifiedIndex"/>`, and `<c:item name="CreationTimeIndex"/>` element values with the customized names created during your initial DynamoDB setup.

2. Save the file.

   The following table describes the preconfigured PingFederate variables in the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl.xml` file.

DynamoDB Client Manager Variables

Variable	Description
`PingFederateSortedOAuthClientId-index`	The name of the OAuth client ID index. This index is used to support sorting by Client ID on the Client Management Admin UI page. This is the default value.
`PingFederateSortedOAuthClientName-index`	The name of the OAuth client name index. This index is used to support sorting by Client Name on the Client Management Admin UI page. This is the default value.
`PingFederateSortedOAuthClientLastModified-index`	The name of the OAuth client last modified index. This index is used to support sorting by Last Modified Time on the Client Management Admin UI page. This is the default value.
`PingFederateSortedOAuthCreationTime-index`	The name of the OAuth client creation time index. This index is used to support sorting by Creation Time on the Client Management Admin UI page. This is the default value.
`EndpointOverride`	An optional endpoint URL which should not be used in production but allows for testing with a local development DynamoDB instance. By default, this value is empty. To test DynamoDB running locally, specify `EndpointOverride` to point to a local endpoint. For example, `<c:item name="EndpointOverride">http://localhost:8000</c:item>`. For more information, see DynamoDB local usage notes in the Amazon DynamoDB documentation.
`dynamoDbBatchSize`	Number of records to request when performing batch operations against DynamoDB. The minimum allowed value is one, the maximum allowed value is 100, and the default value is 50.
`ApiCallTimeout`	The amount of time in milliseconds to allow the client to complete the execution of the API call. The default value is 10000.

Variable	Description
`ApiCallAttemptTimeout`	The amount of time in milliseconds to wait for the HTTP request to complete before giving up and timing out. The default value is 1000.
`query-client-count`	The number of OAuth clients returned on search operations for 'Check Usage' pop-ups of resources referenced by clients. Negative values signify the retrieval of all clients. The default value is 2000.

4. Start or restart the PingFederate service.

> **Note**
>
> For a clustered PingFederate, replicate this new configuration to other engine nodes on **System** > **Server** > **Cluster Management**. Start or restart the PingFederate service on each engine node to active the change.

> **Caution**
>
> Frequent use of certain Admin UI operations, such as searching the **OAuth Client** list or detecting usage on various pages (**Access Token Management**, **OpenID Connect Policy Management**, **Token Exchange Processor Policies**, **CIBA Request Policies**, and **Trusted CAs**), may result in full table scans. This can impact the table's required Read Capacity.

Configuring directories for client storage

Specific schema objects are required in order for PingFederate to store OAuth client records on your directory server. LDIF scripts are provided for supported directory servers.

About this task

> **Caution**
>
> PingFederate does not migrate client records from one storage medium to another. You must recreate your clients after updating the client storage configuration. If you need only a few clients, you can recreate them using the administrative console.
> If you need a large number of clients, use the administrative API to retrieve your client records before updating the client storage. Update the client storage configuration and recreate your clients using the administrative API based on the retrieved records. For more information, see PingFederate administrative API.

Steps

1. Review the LDIF scripts for your directory server provided in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/ldif-scripts` directory.

2. Replace placeholder values with relevant information from your directory server.

3. Run the LDIF scripts to update your LDAP schema.

> **Note**
>
> For Active Directory, run the script to create the attributes, then run the script to create the object class. For PingDirectory, run the **`ldapmodify`** command. For example: `ldapmodify --defaultAdd --filename "<path to ldif file>\oauth-client-management-pingdirectory.ldif" -h <hostname> -p <port> -D "<adminDN>" -w <adminPassword>`

4. If you have not already done so, create an LDAP datastore for your directory server on System > Data & Credential Stores > Data Stores.

5. Copy the system ID of the applicable LDAP datastore from System > Data & Credential Stores > Data Stores.

6. Edit the `<pf_install>/pingfederate/server/default/data/config-store/org.sourceid.oauth20.domain.ClientManagerLdapImpl.xml` file.

   > **Note**
   >
   > For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System > Server > Cluster Management** as explained in later steps.

   1. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your datastore connection.

      Example:

      If the system ID is `LDAP-123456789ABCDEF123456789ABCDEF123456A0AC`, update the configuration file as follows.

      ```
      ...
      <!-- Data store id -->
      <c:item name="PingFederateDSJNDIName">LDAP-123456789ABCDEF123456789ABCDEF123456A0AC</c:item>
      ...
      ```

   2. Enter a value for the `<c:item name="SearchBase"/>` element.

      > **Tip**
      >
      > This is the distinguished name (DN) that points to the client location. For more information, see the inline comment and the LDIF scripts in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/ldif-scripts` directory.

   3. Update the attribute names only if you have changed attribute names in the LDIF scripts located in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/ldif-scripts` directory.

   4. Save the file.

7. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storing OAuth client configuration` section.

```
# Service for storing OAuth client configuration.
# Supported classes are
#   org.sourceid.oauth20.domain.ClientManagerXmlFileImpl  : Use this service-point for an XML
implementation.
#   org.sourceid.oauth20.domain.ClientManagerJdbcImpl     : Use this service-point for a Jdbc
implementation.
#   org.sourceid.oauth20.domain.ClientManagerLdapImpl     : Use this service-point for an LDAP
implementation.
#   org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl : Use this service-point for a
DynamoDB implementation
#   org.sourceid.oauth20.domain.ClientManagerGenericImpl  : Use this service-point if you have
specified a custom ClientStorageManager implementation above.
client.manager=org.sourceid.oauth20.domain.ClientManagerXmlFileImpl
```

2. Change the value of the `client.manager` service to `org.sourceid.oauth20.domain.ClientManagerLdapImpl`.

> **Note**
>
> For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can’t replicate this change to other nodes.

8. Start or restart PingFederate.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

9. In the directory, create indexes for the following OAuth client attributes. If you are using Active Directory, skip this step because the script for Active Directory creates these indexes. If you are using PingDirectory, see Indexing client attributes in PingDirectory for more information.

Attribute name	Index type
`pf-oauth-client-id`	equality
`pf-oauth-client-id`	ordering
`pf-oauth-client-id`	substring
`pf-oauth-client-name`	equality
`pf-oauth-client-name`	ordering
`pf-oauth-client-name`	substring
`pf-oauth-client-last-modified`	ordering

Related links

- Configuring an LDAP connection
- System requirements

Indexing client attributes in PingDirectory

If you use PingDirectory, or another directory, to store OAuth client records for PingFederate, you must index the client attributes.

About this task

Index these OAuth client attributes using the procedure below.

Attribute name	Index type
`pf-oauth-client-id`	equality
`pf-oauth-client-id`	ordering
`pf-oauth-client-id`	substring
`pf-oauth-client-name`	equality
`pf-oauth-client-name`	ordering
`pf-oauth-client-name`	substring
`pf-oauth-client-last-modified`	ordering

Steps

1. Create the indexes using the PingDirectory `dsconfig` utility.

   The `dsconfig` utility is interactive, letting you enter command arguments. The following example creates the three indexes for the `pf-oauth-client-id` attribute.

   ```
   $ bin/dsconfig create-local-db-index \
                    --backend-name userRoot \
                    --index-name pf-oauth-client-id \
                    --set index-type:equality \
                    --set index-type:ordering \
                    --set index-type:substring
   ```

2. After creating the indexes, build them using the `rebuild-index` utility.

   The following example builds the required indexes.

```
$ bin/rebuild-index \
                  --baseDN "dc=example,dc=com" \
                  --index pf-oauth-client-id \
                  --index pf-oauth-client-name \
                  --index pf-oauth-client-last-modified
```

Related links

- PingDirectory Administration Guide: Working with Indexes

Using custom solutions for client storage

Use the PingFederate SDK to implement a custom solution for client storage.

About this task

> **Caution**
>
> PingFederate does not migrate client records from one storage medium to another. You must recreate your clients after updating the client storage configuration. If you need only a few clients, you can recreate them using the administrative console.
> If you need a large number of clients, use the administrative API to retrieve your client records before updating the client storage. Update the client storage configuration and recreate your clients using the administrative API based on the retrieved records. For more information, see PingFederate administrative API.

Steps

1. Implement the `ClientStorageManagerV2` interface.

   This interface includes a `search()` method, allowing developers to provide efficient implementations of the pagination and search functions exposed in the administrative console.

   For more information, see the Javadoc for the `ClientStorageManagerV2` interface, the `SampleClientStorage.java` file for a sample implementation, and the SDK Developer's Guide for build and deployment information.

   > **Tip**
   >
   > The Javadoc for PingFederate and the sample implementation are in the `<pf_install>/pingfederate/sdk` directory.

2. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   1. Go to the `# Service for storing OAuth client configuration` section.

```
# Service for storing OAuth client configuration.
# Supported classes are
#   org.sourceid.oauth20.domain.ClientManagerXmlFileImpl  : Use this service-point for an XML
implementation.
#   org.sourceid.oauth20.domain.ClientManagerJdbcImpl     : Use this service-point for a Jdbc
implementation.
#   org.sourceid.oauth20.domain.ClientManagerLdapImpl     : Use this service-point for an LDAP
implementation.
#   org.sourceid.oauth20.domain.ClientManagerDynamoDBImpl : Use this service-point for a
DynamoDB implementation
#   org.sourceid.oauth20.domain.ClientManagerGenericImpl  : Use this service-point if you have
specified a custom ClientStorageManager implementation above.
client.manager=org.sourceid.oauth20.domain.ClientManagerXmlFileImpl
```

2. Change the value of the `client.manager` service to the name of the class implementing the `ClientStorageManagerV2` interface.

> **Note**
>
> For a clustered PingFederate environment, you must edit the `service-points.conf` file on each node manually because cluster replication can’t replicate this change to other nodes.

3. Start or restart PingFederate.

> **Note**
>
> PingFederate was tested with vendor-specific Java database connectivity (JDBC) 4.2 drivers. Learn more in Compatible database drivers. Start or restart the PingFederate service on each engine node to activate the change.

Account-linking datastores

Configure where you want to store account links, either internally or externally.

When a service provider (SP) is configured to use account linking for an identity provider (IdP) connection, by default PingFederate uses the built-in Hyper SQL Database (HSQLDB) as the account-link repository. You can also configure PingFederate to store account links on an external database server or directory server. For specific instructions on how to configure these options, see the following topics:

- Configuring external databases for account-link storage
- Configuring directories for account-link storage

> **Caution**
>
> Use the built-in HSQLDB only for trial or training environments. For testing and production environments, always use a secured external storage solution for proper functioning in a clustered environment.
> Testing involving HSQLDB is not a valid test. In both testing and production, it might cause various problems due to its limitations and HSQLDB involved cases are not supported by Ping Identity.

Related links

- Account linking
- Server Clustering Guide

Configuring external databases for account-link storage

A specific table is required in order for PingFederate to store account links on your database server. Table-setup scripts are provided for supported database servers.

Steps

1. Create a database for account linking using one of the table-setup scripts located in the `<pf_install>/pingfederate/server/default/conf/account-linking/sql-scripts` directory.
2. Go to System > Data & Credential Stores > Data Stores and create a new datastore to connect PingFederate to the database. For more information, see Configuring a JDBC connection.
3. On the Data Stores window, copy the system ID of the new account-linking datastore.
4. In the `org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory, replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your datastore connection and save the file.

   > **Note**
   >
   > For a clustered PingFederate environment, edit this file on the administrative console node first, and then replicate to other engine nodes using **System** > **Server** > **Cluster Management** as explained in later steps.

   Example:

   For example, if the system ID is `JDBC-123456789ABCDEF123456789ABCDEF123456A0AC`, update the `org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl.xml` file as follows.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <c:config xmlns:c="http://www.sourceid.org/2004/05/config">
       <c:item name="PingFederateDSJNDIName">JDBC-123456789ABCDEF123456789ABCDEF123456A0AC</c:item>
   </c:config>
   ```
5. Start or restart PingFederate.
6. If you are running PingFederate in a cluster, go to System > Server > Cluster Management and replicate this change to other runtime servers.

Related links

- System requirements
- Configuring a JDBC connection

Configuring directories for account-link storage

You can create and configure a directory server to store account linking data.

Before you begin

> **Note**
>
> User accounts for linking must exist in the directory prior to establishing the account link. The Account Linking Service does not add users to the directory server, it only updates `AccountLinkDataAttribute` for a given user.

Steps

1. Go to System > Data & Credential Stores > Data Stores and create a new datastore to connect PingFederate to the directory. For more information, see Configuring an LDAP connection.

2. Copy the system ID of the new account-linking datastore.

3. Edit the `<pf_install>/pingfederate/server/default/conf/service-points.conf` file.

   Locate the service-point for the account linking service.

   ```
   # Service/adapter for storage of account linking
   # Supported classes:
   # org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl : Use this service-point for a
   database implementation
   # org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl : Use this service-point for an
   LDAP implementation
   account.linking.service=org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl
   ```

   Update the `class` value to `org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl`.

4. Edit the `<pf_install>/org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl.xml` file.

   The following example shows the default content of the file.

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <c:config xmlns:c="http://www.sourceid.org/2004/05/config">

      <!-- Data store id -->
      <c:item name="PingFederateDSJNDIName"></c:item>

      <!-- LDAP search base -->
      <c:item name="UserSearchBase"></c:item>

      <!-- LDAP username attribute.  ex: sAMAccountName -->
      <c:item name="UsernameAttribute"></c:item>

      <!-- Attribute on user object to place Account Linking data -->
      <c:item name="AccountLinkDataAttribute"></c:item>

   </c:config>
   ```

   Insert the applicable values between the XML tags as shown in the following table.

Item name	Element value
`PingFederateDSJNDIName`	The system ID of new account-linking datastore.
`UserSearchBase`	The location in the directory server from which the search begins.
`UsernameAttribute`	The attribute that represents the user identifier.
`AccountLinkDataAttribute`	The attribute to store account linking data. **Note** The `AccountLinkDataAttribute` can be any multivalued string attribute on a user object class. We recommend that you extend the LDAP schema with a custom attribute for use here. For more information on extending the Active Directory schema, see Extending the Schema from Microsoft.

5. Start or restart PingFederate.
6. If you are running PingFederate in a cluster, go to System > Server > Cluster Management and replicate this change to the other runtime servers.

   > **Note**
   >
   > You must also manually apply the changes made in the `service-points.conf` file and then start or restart PingFederate on each runtime server.

7. In the directory, create equality indexes on the LDAP attribute types you specified for the configuration properties `UsernameAttribute` and `AccountLinkDataAttribute`.

   Example:

   For example, you would need to create equality indexes on `sAMAccountName` and `AccountLink` if you had specified the following in step 4:

   ```
   <!-- LDAP username attribute. ex: sAMAccountName -->
   <c:item name="UsernameAttribute">sAMAccountName</c:item>

   <!-- Attribute on user object to place Account Linking data -->
   <c:item name="AccountLinkDataAttribute">AccountLink</c:item>
   ```

Related links

- System requirements
- Configuring an LDAP connection

Password Credential Validators

PingFederate provides an authentication mechanism using plugin password credential validators (PCVs). This feature provides centralized credential validation for various PingFederate components and configurations.

To manage Password Credential Validators, go to System > Data & Credential Stores > Password Credential Validators.

For each instance of the HTML Form Adapter, the HTTP Basic Adapter, and the Username Token Processor, you can select the same PCV instance, a unique PCV instance, or multiple PCV instances. When you select multiple PCV instances for a given adapter or token processor instance, if the first PCV instance fails to authenticate a user, the PCV returns control to the adapter or the token processor. The adapter or the token processor then tries the next PCV instance. The cycle stops until a PCV instance succeeds or the last PCV instance also fails.

For OAuth clients using the Resource Owner Password Credentials grant type, you configure a grant-mapping configuration to fulfill the persistent grant contract using the attribute values from the applicable PCV instances.

> **Note**
>
> You can only create one grant-mapping configuration per applicable PCV instance.

If you want to manage OAuth client records using the OAuth Client Management Service or persistent grants using the OAuth Access Grant Management Service, you must select a PCV instance when configuring authorization server settings. When accessing these services, you must include in the requests valid credentials via HTTP Basic authentication scheme.

PingFederate is distributed with the following plugin PCVs.

LDAP Username Password Credential Validator

Validates credentials based on an LDAP look-up in an organization's user-datastore.

PingID PCV (with integrated RADIUS server)

Validates credentials from a VPN RADIUS client based on an LDAP look-up in an organization's user-datastore.

PingOne for Enterprise Directory Password Credential Validator

Validates credentials stored in PingOne for Enterprise Directory.

RADIUS Username Password Credential Validator

Validates credentials based on the RADIUS protocol on an organization's RADIUS server.

Simple Username Password Credential Validator

Validates credentials maintained by PingFederate.

> **Note**
>
> By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
> If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System** > **Server** > **General Settings**.

Choosing a Password Credential Validator

You can choose the type of Password Credential Validator (PCV) that you want to use in PingFederate. You must also specify the PCV's name, ID, and whether it uses a parent instance.

About this task

Available PCV types are determined by plug-in `.jar` files loaded in the `<pf_install>/pingfederate/server/default/deploy` directory. Several validator plugins are bundled with PingFederate. You can add other plugins from the Ping Identity Downloads website.

Steps

1. On the Type tab, enter a name and an ID for the instance.
2. Select the type of the PCV from the Type list.
3. Optional: Select a Parent Instance from the list.Use this option when creating an instance that is similar to an existing one. The child instance inherits the configuration of its parent. You can also override one or more settings during the setup. Select the Override ... check box and make the adjustments as needed in one or more subsequent windows.

Password Credential Validator instance configurations

The instance configuration of a Password Credential Validator (PCV) varies depending on the credential validators deployed on your server.

For PCVs bundled with PingFederate, see the following topics:

- Configuring the LDAP Username Password Credential Validator
- Configuring the PingOne for Enterprise Directory Password Credential Validator
- Configuring the RADIUS Username Password Credential Validator
- Configuring the Simple Username Password Credential Validator

Related links

- Integrate PingID with your VPN

Configuring the LDAP Username Password Credential Validator

The LDAP Username password credential validator (PCV) verifies credentials using an organization's LDAP datastore.

About this task

When an authentication error occurs, PingFederate automatically parses the messages returned by PingDirectory, Microsoft Active Directory (AD), Oracle Unified Directory (OUD), or Oracle Directory Server (ODS) and categorizes them with error conditions.

When validating against a directory server other than PingDirectory, AD, OUD, or ODS, administrators can define custom message categorization by mapping specific error messages with wildcard support to the desired error conditions.

The error messages are returned to the HTML Form Adapter instances and the OAuth clients using the Resource Owner Password Credential grant type. The HTML Form Adapter is designed to show the error message it receives from the LDAP Username PCV. OAuth-client developers can create custom experiences based on the error responses, which contain the error messages. The HTML Form Adapter uses the relevant error conditions to determine the LDAP password-change scenarios and to present the relevant messages to the end users.

> **Tip**
>
> These customizable messages are stored in the PingFederate message file, `pingfederate-messages.properties`, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.
> You can localize these messages by using the PingFederate localization framework for an international audience. For more information, see Localizing messages for end users.

Steps

1. Go to the Instance Configuration tab.
2. Optional: Override the authentication error messages.

   > **Note**
   >
   > You might require this option in order for a directory server other than PingDirectory, AD, OUD, or ODS to support the password change function in the HTML Form Adapter or to alter the end-user messages associated with that function.

   1. Click Add a new row to 'Authentication Error Overrides'.
   2. Enter an applicable LDAP error message under Match Expression.

      > **Tip**
      >
      > You can use wildcard asterisks to match messages returned from your directory server. For example, `*expired*`.

   3. Select a relevant error condition from the Error list.
   4. Optional: Enter a key name or an error message under Message Properties Key.

      No value

      If you skip this field, PingFederate returns the default message based on the selected error condition.

      A unique key name

      If you enter a key name in this field and then add the key name with a key value (the desired error message) to the PingFederate message file, PingFederate returns that key value.

      The key name must be unique. Furthermore, you may localize these messages by using the PingFederate localization framework for an international audience.

      An error message

      If you enter an error message in this field (without defining it in the PingFederate message file), PingFederate returns your message verbatim.

5. Click Update under Action.

6. Repeat these steps to add more overrides as needed.

> **Note**
>
> Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.
> Use the **up** and **down** arrows to change the display order. The display order does not affect runtime processing.

3. Select the LDAP datastore and enter information into the required fields.

   For more information about each field, see the following table.

Field	Description
LDAP Datastore (Required)	The LDAP datastore configured in PingFederate. If you have not yet configured the server to communicate with the directory server you need, click Manage Data Stores. + **Note** When connecting to an AD LDAP server, if you want to enable the password changes, password reset, or account unlock features in the HTML Form Adapter, you must secure the datastore connection to your AD LDAP server using LDAPS. AD requires this level of security to allow password changes.
Search Base (Required)	The location in the directory server from which the search begins.

Field	Description
Search Filter (Required)	The LDAP query to locate a user record. If your use case requires the flexibility of allowing users to identify themselves using different attributes, you can include these attributes in your query. For instance, the following search filter allows users to sign on using either the `sAMAccountName` or `employeeNumber` attribute value through the HTML Form Adapter. `(|(sAMAccountName=${username})(employeeNumber=${username}))` **Important** To ensure that your service providers (SPs) always get the expected attribute, select a specific user attribute as the source of the subject identifier when configuring the applicable SP connections. There are several ways to do so: ◦ Extend the PCV contract and fulfill the subject identifier through the HTML Form Adapter. For more information, see Extending the contract for the credential validator. ◦ Add a data source in the SP connection and fulfill the subject identifier through a datastore query. For more information, see Configuring attribute sources and user lookup. ◦ If you use authentication policy in conjunction with a policy contract, you can add a data source in the contract mapping configuration and fulfill the subject identifier in an SP connection through the authentication policy contract. For more information, see Applying policy contracts or identity profiles to authentication policies. When configuring multifactor authentication using PingID, where you chain an instance of the PingID Adapter behind an HTML Form Adapter instance, ensure that you also select a specific user attribute as the incoming user attribute for the PingID Adapter instance. For example, if you have set up PingFederate as the identity bridge for your PingOne for Enterprise account and have selected `sAMAccountName` as the subject identifier in the SP connection, you should also select `sAMAccountName` as the incoming user attribute for your PingID Adapter instance. You can accomplish this through an instance of the Composite Adapter or an authentication policy. For more information, see **Input User ID Mapping** in Configuring a Composite Adapter instance or **Incoming User ID** in Specifying incoming user IDs, respectively.
Scope of Search	The level of search to perform in the search base. One Level indicates a search of objects immediately subordinate to the base object, not including the base object itself. Subtree indicates a search of the base object and the entire subtree within the base object distinguished name. The deault selection is Subtree.
Case-Sensitive Matching	The option to enable case-sensitive matching between the LDAP error messages returned from the directory server and the Match Expression values specified on this window. This check box is selected by default.

Field	Description
Advanced fields for self-service password reset, account unlock, and user name recovery through the HTML Form Adapter	
Display Name Attribute	The LDAP attribute used for personalizing messages to the users. This field is applicable for all password reset types (other than None), account unlock, and user name recovery. The default value is `displayName`.
Mail Attribute (for password reset)	The LDAP attribute containing the email address of the users. This field is required when password reset using one-time link or one-time password is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance. **Note** When configuring in conjunction with user name recovery, this attribute should correspond to the attribute specified on the left side of the **Mail Search Filter** field. The default value is `mail`.
SMS Attribute (for password reset)	The LDAP attribute containing the telephone number of the users. This field is required when password reset using text message is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance. This field has no default value.
PingID Username Attribute (for password reset)	The LDAP attribute containing the PingID user name of the users. This field is required when password reset using PingID is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance.
Mail Search Filter (for user name recovery)	The LDAP query to locate a user record using an email address, such as `mail=${mail}`. This field is required when user name recovery is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance. **Note** When configuring in conjunction with password reset, the attribute specified on the left side of this search filter should correspond to the attribute specified in the **Mail Attribute** field.

Field	Description
Username Attribute (for user name recovery)	The LDAP attribute containing the user identifier of the users. This field is required when user name recovery is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance. ⓘ **Note** This attribute should correspond to the attribute specified on the left side of the **Search Filter** field.
Mail Verified Attribute (for user name recovery)	The LDAP attribute indicating whether the user's email address is verified. The expected value of this user attribute is either `true` or `false` (case insensitive). This field is required when user name recovery using only verified email addresses is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance.
Enable PingDirectory Detailed Password Policy Requirement Messaging (for change password and password reset)	For password changes through an LDAP PCV backed by PingDirectory, this option indicates whether PingDirectory should return detailed password policy violations to PingFederate so that the invoking authentication source, namely the HTML Form Adapter, can present them to the end-users. **Important** The **User DN** associated with the selected LDAP Datastore must be granted the proper access control instruction (ACI) in PingDirectory. For more information, see Configuring the password validation details request control ACI.
Expect Password Expired Control (for password expiry)	When selected, PingFederate expects the LDAP datastore to return the password expired control when a user presents valid credentials but their password has expired. PingFederate can use this control in the bind response to distinguish between the following cases: ◦ An expired password with valid credentials ◦ An expired password with invalid credentials By default, the checkbox is cleared. When using PingDirectory as the credential store, if a user's password has expired, the following settings ensure PingFederate doesn't direct the user to the Change Password form when they enter an invalid password: ◦ Select the Expect Password Expired Control checkbox. ◦ Set the return-password-expiration-controls setting in the PingDirectory password policy to always.

Related links

- Configuring an HTML Form Adapter instance

Configuring the PingOne for Enterprise Directory Password Credential Validator

The PingOne for Enterprise Directory Username Password Credential Validator (PCV) verifies credentials stored in your PingOne for Enterprise Directory.

Before you begin

To use the PingOne for Enterprise PCV, you must have:

- A PingOne for Enterprise account
- A PingFederate account

For more information, see Managing PingOne for Enterprise Directory users in the PingOne for Enterprise documentation.

Steps

1. On the Instance Configuration tab, enter your account information in Client ID and Client Secret.

   For more information about each field, refer to the following table. All fields are required.

Field	Description
Client ID	The REST API client ID is a unique identifier PingFederate uses to identify itself to the PingOne for Enterprise Directory API. For more information, see View or renew directory API credentials in the PingOne for Enterprise documentation.
Client Secret	The client secret is used to authenticate the client ID against the PingOne for Enterprise Directory API. For more information, see View or renew directory API credentials in the PingOne for Enterprise documentation.
Advanced Fields	
PingOne URL	The PingOne for Enterprise Directory API. The default value is https://directory-api.pingone.com/api.
Authenticate by Subject URL	The relative path for user authentication. The default value is `/directory/users/authenticate?by=subject`.
Reset Password URL	The relative path for password reset. The default value is `/directory/users/password-reset`.
SCIM User URL	The relative path for searching users requesting password reset. The default value is `/directory/user`.
Connection Pool Size	The maximum size of the connection pool to PingOne for Enterprise Directory. The default value is `100`.

Field	Description
Connection Pool Idle Timeout	The maximum time (in milliseconds) that a connection can remain idle before it is closed and removed from the connection pool. The default value is `4000`.

Configuring the RADIUS Username Password Credential Validator

The RADIUS Username Password Credential Validator verifies credentials using the RADIUS protocol.

About this task

RADIUS supports strong authentication with both one-step (a combination of regular password and a one-time password in one field) and two-step (challenge-response) authentication. Two-step authentication is supported in the HTML Form Adapter.

> **Important**
>
> If your RADIUS server is a Microsoft Network Policy Server (NPS), passwords containing special characters will not be encoded and decoded properly due to limitations with NPS.

> **Tip**
>
> RADIUS server messages are used by the HTML Form Adapter to determine the two-step authentication scenarios and to present a sign on window to the end users.

Steps

1. On the Instance Configuration tab, configure one or more RADIUS servers.
    1. Click Add a new row to 'RADIUS Servers'.
    2. In each field, enter the required information.

       For more information about each field, refer to the following table. All fields are required.

Field	Description
Hostname	The IP address of the RADIUS server. For failover, enter one or more backup RADIUS servers by adding each server in its own row of the table. Each row represents a distinct RADIUS server that can be used for failover. PingFederate attempts to make a connection to each server in the order listed until a successful connection is obtained.
Authentication Port	The UDP port used to authenticate to the RADIUS server. The default value is `1812`.

Field	Description
Authentication Protocol	The protocol used to authenticate to the RADIUS server. The available choices are Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP). Select the protocol expected by your RADIUS server. The default selection is PAP.
Shared Secret	The password shared between PingFederate and the RADIUS server used to encrypt the attribute identifying the NAS (Network Access Server) originating the request for access.

> **Note**
>
> The NAS-IP-Address attribute is added to all Access-Request packets sent to the RADIUS server. The value is copied from the `pf.engine.bind.address` property in the `<pf_install>/pingfederate/bin/run.properties` file. Only IPv4 addresses are supported.

3. Click Update in the Action column.

4. Repeat these steps to add more RADIUS servers as needed.

> **Note**
>
> Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.
> Use the up and down arrows to adjust the order in which you want PingFederate to attempt credential authentication. If an earlier RADIUS server fails to validate the credentials, PingFederate moves sequentially through the list until credential validation succeeds. If none of the RADIUS servers is able to authenticate the user's credentials, the credential validation process fails.

2. Optional: Click Show Advanced Fields to reconfigure default settings.

For more information about each field, refer to the following table. All fields are required.

Field	Description
NAS Identifier	The password shared between PingFederate and the RADIUS server used to encrypt the attribute identifying the NAS (Network Access Server) originating the request for access. The default value is `PingFederate`.
Timeout	The maximum number of milliseconds before a connection timeout to the RADIUS server. The default value is `3000`.
Retry Count	The number of times to retry a failed connection before moving to the next host. The default value is `3`.

Configuring the Simple Username Password Credential Validator

The Simple Username Password Credential Validator verifies credentials maintained by PingFederate. This validator is best used for testing purposes or for an organization with few accounts.

Steps

1. On the Instance Configuration tab, click Add a new row to 'Users'.
2. Enter a user name, followed by a password (twice).
3. Click Update in the Action column.
4. Repeat these steps to add more user credentials as needed.

> **Note**
>
> Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.
> Use the up and down arrows to adjust the order in which you want PingFederate to attempt credential authentication. PingFederate moves sequentially through the list until credential validation succeeds or no match is found.

Extending the contract for the credential validator

You can extend Password Credential Validator (PCV) instance contracts to return attribute values relevant to authenticated users.

About this task

In some use cases, you might want to extend the contracts of the PCV instance. For example, you might use extended attributes to map into a `USER_KEY` for an OAuth persistent grant configuration.

This capability allows the validator to return attribute values pertaining to the authenticated users from PingOne for Enterprise Directory, a directory server, or a RADIUS server.

> **Tip**
>
> If you are configuring an HTML Form Adapter instance with an instance of the LDAP Username Password Credential Validator, extend the contract of the adapter by the same attribute names in order for the credential validator to pass extended attribute values to the HTML Form Adapter instance.
> If you are configuring the HTML Form Adapter instance with an instance of the RADIUS Username Password Credential Validator, you only need to extend the contract of the HTML Form Adapter instance itself.

Steps

1. Copy the vendor-specific attribute dictionaries into the `pingfederate/server/default/conf/radius` directory.

> **Note**
>
> The format of the dictionaries must use the FreeRadius dictionary syntax.

2. Edit the existing `dictionary` file to include each of the dictonaries.

3. Optional: On the Extended Contract tab, enter an attribute name and click Add.

> **Note**
>
> Click **Edit**, **Update**, or **Cancel** to make or undo a change to an existing entry. Click **Delete** or **Undelete** to remove an existing entry or cancel the removal request.

Finishing the Password Credential Validator instance configuration

On the Summary tab, you can review your configuration.

Steps

1.
   - To keep your changes, click Save.
   - To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
   - To discard your changes, click Cancel.

Active Directory and Kerberos

You can configure PingFederate to authenticate users through the following identity provider (IdP) adapters or token processors.

Adapter or Token Processor	Description
PingFederate integrated Kerberos Adapter	Using the built-in Kerberos Adapter with a configured AD domain allows a PingFederate identity provider (IdP) server to perform single sign-on (SSO) to service provider (SP) applications based on Kerberos tickets.
PingFederate integrated Kerberos Token Processor	The built-in Kerberos Token Processor accepts and validates Kerberos tokens through a configured Kerberos Realm from a web service client.

> **Important**
>
> As of version 10.3 and above, PingFederate no longer supports the IWA integration kit. For more information about Migrating from the IWA Integration Kit to the PingFederate Kerberos adapter, see https://support.pingidentity.com/s/article/Migrating-from-the-Integrated-Windows-Authentication-integration-kit-to-the-PingFederate-Kerberos-adapter in the Ping Identity Support Portal.

Configuring Active Directory domains or Kerberos realms

You can configure an Active Directory (AD) domain or Kerberos realm to authenticate users.

Steps

1. Go to System > Data & Credential Stores > Active Directory Domains/Kerberos Realms.
2. From the Manage AD Domains/Kerberos Realms window, configure the AD environment to integrate with PingFederate. For more information, see Configuring the Active Directory environment.
3. Click Add Domain/Realm to create an AD domain.

> **Important**
>
> Do not configure subdomains if the parent domain in the same forest is already configured. For more information, see Multiple-domain support.

> **Note**
>
> Click the name of an existing domain to edit it. Use the **Delete** and **Undelete** links to remove a domain or cancel a removal request.

Multiple-domain support

If your network uses multiple domains in a single server forest, you can configure one domain within PingFederate if there is a trust relationship with the other domains you want to use.

This configuration requires a trust relationship among domains, which is established by default when subdomains or separate domains are created within the same forest. For more information, see https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc773178(v=ws.10)?redirectedfrom=MSDN.

> **Note**
>
> If you are configuring only one domain, then you also need to configure only one Service Principal Name. For more information, see Configuring the Active Directory environment.

If your network topology consists of multiple forests without a trust relationship between them, you must configure multiple adapter or token processor instances. Map each instance to a separate domain and then map these adapter or token processor instances to your service provider (SP) connections that authenticate using the integrated Kerberos Adapter or the integrated Kerberos Token Processor.

For information about configuring Kerberos authentication for multiple-domain Active Directory trusts, see https://support.pingidentity.com/s/article/How-to-configure-IWA-with-multiple-Active-Directory-trusts in the Ping Identity Support Portal.

Configuring the Active Directory environment

You can configure Active Directory to access a domain and enable Kerberos as an authentication option for it.

About this task

To enable Kerberos authentication, you must make several Active Directory configuration changes to grant PingFederate access to the domain and add the domain to PingFederate.

> **Important**
>
> Do not configure subdomains if the parent domain in the same forest is already configured. For more information, see Multiple-domain support.

> **Note**
>
> You must have Domain Administrator permissions to make the required changes.

Steps

1. Create a domain user account that PingFederate can use to contact the Kerberos Key Distribution Center (KDC). The account should belong to the Domain Users group. We recommend that you set the password with no expiration.
2. Use the Windows utility `setspn` to register Service Principal Name (SPN) directory properties for the account by executing the following command on the domain controller. `setspn -s HTTP/<pf-idp.domain.name> <pf-server-account-name>` , where <*pf-idp.domain.name*> is the canonical name of the PingFederate server and <*pf-server-account-name*> is the domain account you want to use for Kerberos authentication. For more information on canonical name, see https://tools.ietf.org/html/rfc2181#section-10.

   > **Note**
   >
   > When executing the `setspn` command, you must capitalize `HTTP` and follow it with a forward slash (`/`).

3. Verify that the registration was successful by executing the following command. `setspn -l <pf-server-account-name>`

   This gives you a list of SPNs for the account. Verify that `HTTP/<pf-idp.domain.name>` is one of them.

   > **Note**
   >
   > After making an SPN change, any authenticated end users must re-authenticate by closing the browser or signing off and back on before attempting single sign-on (SSO).

Adding Active Directory domains and Kerberos realms

You can configure Active Directory domains or Kerberos realms that PingFederate uses to contact the domain controllers or the key distribution centers (KDCs) for verifying user authentication.

About this task

The procedure for adding an Active Directory domain or Kerberos realm depends on whether PingFederate is deployed on-premise or in a cloud. Perform the procedure on one of the following tabs.

PingFederate on-premise

Adding domains and realms when PingFederate is on-premise

When PingFederate is deployed on-premise, use the following procedure.

Steps

1. In the Manage Domain/Realm window, configure the following settings.

Field	Description
Connection Type	Select Directly.
Domain/Realm Name	The fully-qualified domain or realm name. For example, companydomain.com.
Domain/Realm Username	The ID for the domain or realm account name.
Domain/Realm Password	The password for the domain or realm account.
Retain Previous Keys on Password Change	Select this check box and click Save to avoid locking out end users with existing Kerberos tickets when the service account password is updated. PingFederate retains each previous key for the period specified in the Key Set Retention Period field on the Manage Domain/Realm Settings tab of the Active Directory Domains/Kerberos Realms window. The default period is 610 minutes. For more information, see Managing domain connectivity settings. **Tip** To clear the previous keys from PingFederate, clear the check box and click **Save**. This check box is selected by default.
Domain Controller/Key Distribution Center Host Names (optional)	Specify the host name or IP address of your domain controller or KDC, such as `dc01-yvr`, and then click Add. Repeat this step to add multiple servers. If a host name is used, PingFederate appends the domain to the host name to formulate the fully qualified domain name (FQDN) of the server unless the Suppress DC/Domain Concatenation check box is selected. If unspecified, PingFederate uses a DNS lookup.
Suppress DC/Domain Concatenation	Select this check box to specify the desired FQDNs under Domain Controller/ Key Distribution Center Host Names. When selected, PingFederate does not append the domain to the host names.

Field	Description
Test Domain/Realm Connectivity	Tests access to the domain controller or KDC from the administrative-console server. When a connection to any of the configured controllers or KDCs is successful, the message `Test Successful` appears. Otherwise, the test returns error messages near the top of the window. 💡 **Tip** For help resolving connectivity issues, select the **Debug Log Output** check box on the **Manage Domain/Realm Settings** window, run the test again, and review the debug messages in the PingFederate server log. This test stops at the first successful result when multiple domain controllers or KDCs are specified, so not all servers are necessarily verified. Depending on the network architecture, the engine nodes deployed in a cluster might establish connections differently. As a result, the engine nodes and the console node might connect to different domain controllers or KDCs.

2. Click Save.

PingFederate in a cloud

Adding domains and realms when PingFederate is in a cloud

When PingFederate is deployed in a cloud, use the following procedure.

Before you begin

Ensure that PingFederate has a connection to a PingOne LDAP Gateway. For more information, see Gateways in the documentation for the PingOne Cloud Platform.

Steps

1. In the Manage Domain/Realm window, configure the following settings.

Field	Description
Connection Type	Select Through PingOne LDAP Gateway.
Domain/Realm Name	The fully-qualified domain or realm name. For example, companydomain.com.
PingOne LDAP Gateway Data Store	Select the datastore that was configured for the PingOne LDAP Gateway.
Test Domain/Realm Connectivity	Tests access to the domain controller or KDC from the administrative console server. When a connection to the configured PingOne LDAP Gateway is successful, the message `Test Successful` appears. Otherwise, the test returns error messages near the top of the window.

2. Click Save.

Managing domain connectivity settings

You can change the default security and logging settings for all configured Active Directory domains and Kerberos realms.

Steps

- On the Manage Domain/Realm Settings tab, change the default transport protocol, the debug option, the timeout value, and the number of retry attempts. For more information, refer to the following table.

Field	Description
Force TCP	When selected, requires use of the Transmission Control Protocol instead of the default User Datagram Protocol. Use this option when firewall or network configurations require acknowledgment that packets are properly received. ⓘ **Note** If you choose this option, you must restart PingFederate after saving the configuration.
Debug Log Output	When selected, sends verbose messages to the PingFederate server log for all interactions with the domain controllers or the Key Distribution Centers (KDCs).
AD Domain Controller/Key Distribution Center Timeout (secs)	The number of seconds that PingFederate waits for a network response from a domain controller or KDC. The default is `3`. ⓘ **Note** This value applies to each attempt PingFederate makes to contact the domain controller or KDC. The new timeout takes effect only after you save the configuration and restart PingFederate..
AD Domain Controller/Key Distribution Center Retries	Specifies the number of times PingFederate tries contacting the domain controller or KDC. The default is `3`.
Key Set Retention Period (mins)	The number of minutes that PingFederate retains the encryption keys associated with the previous password of the Kerberos service account. The allowed range is 0 to 10080 minutes (seven days) and the default value is 610 minutes. PingFederate only retains the key sets associated with previous passwords if you select the Retain Previous Keys on Password Change check box on the Manage Domain/Realm window. For more information, see Adding Active Directory domains and Kerberos realms.

External systems

Many use cases require communications between PingFederate and other systems, such as PingOne or a database server. System > External Systems is where you can set up and maintain such integrations.

Connections to PingOne

Some PingFederate plugins that access services provided by PingOne can leverage a secure connection between PingFederate and PingOne.

For example, when configuring instances of PingFederate's PingOne MFA and PingOne Protect plugins, you can select a connection and one of the connection's PingOne environments. You do not need to enter credentials and other connection details for each instance. Also, you do not need to update each plugin instance individually when, for example, you replace the connection's credential.

The central element of a connection is its credential, which an administrator generates on PingOne and copies to PingFederate when they create the connection. During runtime, a PingFederate plugin instance that was configured to use the connection gets the credential and uses it to access a PingOne environment.

Administrators with access to both PingFederate and PingOne create the connection. For more information, see Creating connections to PingOne.

You can create multiple connections but most deployments require only one enabled connection. Typically, you use only one connection from a PingFederate cluster to a given PingOne organization. A single connection can be configured to provide access to multiple environments in PingOne.

Cards on PingFederate's System > External Systems > PingOne Connections window display information about each connection to PingOne. When the card is collapsed, it shows only the connection's name, ID, and status toggle switch. Use the toggle switch to enable and disable the connection on PingFederate.

> **Note**
>
> Administrators can disable a connection on PingFederate or PingOne. Plugins can use a connection only if it is enabled on both PingFederate and PingOne.

When a connection card is expanded, it shows more information about the connection on tabs, including the status of the main connection and the plugin connections, if any.

Tab	Description
Summary	In the Summary tab's Summary Details section, the Description and Created time come from PingFederate. The Connection ID in PingOne, Credential ID, Organization Name, and Region come from the connection credential. In the Summary tab's Configuration section, the image shows: • the status of the connection between PingFederate and PingOne • which plugin types can use this kind of connection and are installed on PingFederate, if any • which service on PingOne each of those plugin types is designed to use • whether any instances of those plugin types are configured to use this particular connection
Usage	The Usage tab lists each plugin instance on PingFederate that is configured to use this connection. The instances are grouped by plugin category.
Environments	The Environments tab shows which environments on PingOne the connection is able to access. You assign environments to connections in PingOne. In PingFederate, when you configure a plugin to use a connection, you select the connection and one of its environments.

Creating connections to PingOne

Administrators can create connections between PingFederate and PingOne that some PingFederate plugins can use to access services provided by PingOne.

About this task

In the following procedure, you create a connection profile and credential in PingOne. Then you use the credential in PingFederate to establish the connection. So you need administrative access to both applications. NOTE: You cannot use the same credential in multiple extant connections.

After creating the connection, you can use it to configure PingFederate plugins that support connections to PingOne, such as the PingOne MFA and PingOne Protect plugins.

For more information about these connections, see Connections to PingOne in the PingFederate documentation and PingFederate connections in the PingOne documentation.

Steps

1. In PingOne, create a connection profile and credential:

    1. Go to Connections > Product Platform > PingFederate.

    2. Click + Add Connection.

    3. On the Create Connection Profile window, enter a Connection Name and Description.

    4. Click Save and Continue.

    5. On the Establish Connection with PingFederate window, copy the connection credential by clickingCopy to Clipboard.

    6. If you will not perform step 2 now, paste and save the credential in a text file because this is the only time you can view and copy the credential.

2. In PingFederate, establish the connection:

    1. Go to System > External Systems > PingOne Connections.

    2. On the PingOne Connections window, click + Add Connection.

    3. On the Add Connection window, paste the credential that was created in step 1.

    4. Enter a new Connection Name or keep the auto-populated name, which is the name of your organization on PingOne. Ensure the name is unique.

    5. Enter a Connection Description.

    6. Click Save. The PingOne Connections window shows the new connection. The connection is enabled by default.

3. Optional: Configure the PingOne unified admin icon in the navigation menu to access your PingOne admin console from PingFederate.

    1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.

    2. Modify the `pf.pingone.admin.url.region` and `pf.pingone.admin.url.environment.id` properties for your environment.

For more information on these properties, see Configuring PingFederate properties.

1. Restart PingFederate.

Next steps

After you create a connection, you can:

- use the connection to configure PingFederate plugins that support these kinds of connections
- change which environments in PingOne the connection can access
- replace or revoke the connection's credentials
- disable the connection
- change the connection's name or description

For more information, see Modifying connections to PingOne.

Modifying connections to PingOne

Administrators can modify a connection between PingFederate and PingOne by, for example, adding a PingOne environment to it, replacing its credential, changing its description, or disabling it

> **Note**
>
> If you have multiple connections to PingOne, to avoid confusion, you can identify a connection by its connection ID in both PingFederate and PingOne. The connection ID never changes, whereas its name can be changed.

Editing connection names and descriptions

You can change the names and descriptions of connections between PingFederate and PingOne.

About this task

The following procedure describes how to change the name and description of the connection in PingFederate. You can also change them in PingOne.

> **Note**
>
> Changing the name and description in PingFederate does not change them in PingOne, and vice versa. You cannot change a connection's ID.

Steps

1. In PingFederate, go to System > External Systems > PingOne Connections.
2. On the PingOne Connections window, open the connection's card by clicking its expand icon.
3. Click the pencil icon.

4. Change the Connection Name, Connection Description, or both.
5. Click Save.

Disabling and enabling connections

You can disable a connection between PingFederate and PingOne in either of those applications; however, to enable a connection, it must be enabled in both applications.

Steps

1. To disable or enable a connection in PingFederate:
    1. Go to System > External Systems > PingOne Connections.
    2. On the connection's card, click the toggle switch to change the connection's state.
2. To disable or enable a connection in PingOne:
    1. Go to Connections > Product Platform > PingFederate.
    2. On the connection's card, click the toggle switch to change the connection's state.

Replacing connection credentials

You can replace or revoke the credential of a connection between PingFederate and PingOne.

About this task

The following procedure describes how to replace a connection's credential, which requires administrative access to both applications.

Steps

1. In PingOne:
    1. Go to Connections > Product Platform > PingFederate.
    2. On the PingFederate Connections window, open the connection's card and go to the Configuration tab.
    3. Click the pencil icon.
    4. Click + Add.

       Result:PingOne generates a new credential, adds it to the tab, and opens a dialog box.
    5. In the dialog box, click the credential to save it to the clipboard.
    6. Close the dialog box.

7. If you will not perform step 2 now, paste and save the credential in a text file because this is the only time you can view and copy the credential.

2. In PingFederate:

    1. Go to System > External Systems > PingOne Connections.

    2. On the PingOne Connections window, open the connection's card.

    3. Click the pencil icon.

    4. Paste the new credential that you created in step 1 in the Credential field.

    5. Click Save.

Next steps

After you replace a connection's credential, you can go to PingOne and revoke the old credential.

Modifying which environments connections can access

PingOne administrators can specify which environments in PingOne a connection can access.

In PingOne, environments are assigned to roles and roles are assigned to connections. When a connection is created, PingOne assigns it default roles and environments. A connection can access all the environments assigned to its roles. For more information, see Edit a connection: Add roles and responsibilities in the documentation for PingOne.

Connections to PingOne for Enterprise

Integrating PingOne for Enterprise with PingFederate provides a powerful solution combining the benefits of an on-premise deployment with the flexibility of a cloud solution.

Steps

1. Go to System > External Systems > Connect to PingOne for Enterprise.

2. On the Connect to PingOne for Enterprise window, click Sign on to PingOne to get your activation key.

3. Sign on to PingOne for Enterprise using your PingOne Admin Portal credentials.

    > **Tip**
    >
    > If you do not have an account, you can register for a free trial.

4. Copy the Activation Key value from the PingOne admin portal.

5. Return to the PingFederate administrative console.

6. On the PingOne Account tab, paste the key value in the Activation Key field. Click Next.

Configuring identity repository settings

Set up a customized directory configure identity repository settings either immediately or at a later time.

About this task

On the Identities tab, you can optionally connect to a directory server.

Steps

- Go to System > External Systems > Connect to PingOne for Enterprise and access the Identities tab.
- To enable directory integration, select Yes, Connect a Directory Server.

  You can create a new datastore or reuse an existing datastore in this configuration.

  ### *Create a new datastore*

  Provide the required information to connect to a directory server, and then click Next.More information about each field is provided in the following table.

Field	Description
Directory Type	Select the type of directory server from the list. See System requirements for a list of supported directory servers.
Data Store Name	Enter the name of the datastore.
Hostname	Enter the location of the directory server. It can be the IP address, the host name, or the fully qualified domain name of the directory server. The entry might include a port number.
Service Account DN	Enter the distinguished name (DN) of the service account that PingFederate can use to communicate with the directory server.
Password	Enter the password associated with the service account.
Search Base	Enter the DN of the location in the directory where PingFederate begins its datastore queries.
Search Filter	Enter the LDAP query to locate a user record for attribute lookup and potentially credential validation. The default value is either `sAMAccountName=${username}` or `uid=${username}`, depending on the selected directory type. If you require a more advanced search filter, ensure the value is a valid LDAP filter. For more information, consult your directory administrators.

When you click Next, PingFederate tries to establish a secure (LDAPS) connection to the directory server.If the directory server does not support LDAPS, the Unsecure Connection window appears. If you want to continue without a secure connection, click Next. Alternatively, you can go back to the Identities tab and specify a different directory server.If the certificate presented by the directory server is not trusted by PingFederate, the Certificate Error window appears. You can import the certificate used by the directory server to establish a secure connection, and then click Next in theIdentities tab and specify a different directory server.

Use an existing datastore

Click Begin, and then follow the on-screen instructions to create a service provider (SP) connection to PingOne for Enterprise.

- Optional: To set up a directory later, select No, Don’t Connect a Directory Server and then click Next.

> **Tip**
>
> This setup scenario is suitable for proof of concept. Multiple local test accounts are created as a result.

Use Cases

On the Use Cases tab, select what you plan to use PingFederate to do.

PingOne SSO

Select this option and click Next if you want to enable single sign-on with PingOne for Enterprise and your configured identity store. For more information about using PingOne SSO, see the *PingOne for Enterprise Administration Guide*.

> **Note**
>
> If you have not connected to an LDAP directory, the system configures test user accounts.

Optionally, you can enable Kerberos authentication (if you are using Active Directory). You can also enable user provisioning. To do this, select Additional SSO Features, and click Begin. After you have finished the Kerberos and provisioning configuration, you will be returned to the Use Cases tab.

PingID VPN (RADIUS)

Select this option and click Next if you want to configure PingFederate to integrate with a RADIUS server to support PingID multifactor authentication through your virtual private network. When you select this option, a Begin button appears. Click Begin.

For more information about using PingID VPN, see the *PingID Administration Guide*.

Configuring the RADIUS server to integrate PingID with your VPN

On the Basic Settings tab, you enable the built-in RADIUS server to integrate PingID with your VPN.

Steps

1. To configure the RADIUS client and server information, fill out the required fields.

   For information about each field, see the following table.

Field	Description
Client IP	Enter the IP address of the VPN server.
Client Shared Secret	Enter the password shared between the VPN server and PingFederate.
Server Authentication Port	Enter the listening port of the integrated RADIUS server. The default value is `1812`.

2. If you are connecting PingFederate to a directory server, select the Validate with LDAP check box to enable first-factor credential validation against the directory server. This check box is selected by default.
3. If you are connecting PingFederate to a directory server and have updated the Search Filter value to compare user input against another user attribute, enter that user attribute name in the PingID Username Attribute field. Click Next.

   Example:

   If you have updated the Search Filter value from `uid=${username}` to `mail=${username}`, enter `mail` in the PingID Username Attribute field.

Configuring provisioning to PingID

If you have chosen to connect PingFederate to a directory server, the Provisioning tab becomes available.

About this task

On the Provisioning tab, you have the option to enable provisioning of users from your directory server to PingID. In this configuration, specify the group where PingFederate should look for member users and update PingID when their email address, first name, or last name has changed.

> **Note**
>
> This provisioning capability is designed to manage existing PingID accounts. It does not create new PingID users. When PingFederate detects that a user has been removed from the specified group or disabled in the directory server, PingFederate sends an update to PingID to disable the PingID service for that account.

Steps

1. Select the Configure Provisioning check box.
2. In the PingID Group DN field, enter the distinguished name (DN) of the applicable group.

   > **Note**
   >
   > The specified group must reside under the hierarchy of the previously-defined **Search Base** value.

3. If you want PingFederate to monitor changes for users through nested group membership, select the Nested check box. Click Next.

Reviewing the PingID VPN (RADIUS) configuration

On the Summary tab, review your configuration.

Steps

1. Perform one of the following actions.

   Choose from:

   - To edit your configuration, click the corresponding tab and follow the wizard.
   - To keep your changes, click Done and continue with the configuration.
   - To discard your changes, click Cancel.

Confirmation

The Confirmation tab displays a summary of the configuration that will be applied to PingFederate.

Click Next to apply the configuration.

Complete

The Complete! tab congratulates you on successfully setting up PingFederate.

Steps

1. Make a note of the instructions under What's Next? You will need to complete these tasks when you begin configuring PingFederate.
2. Click Done.

Managing PingOne for Enterprise settings

You can configure PingOne for Enterprise's general settings and single sign-on (SSO) settings, and enable and configure a RADIUS server to integrate PingID with a VPN.

About this task

Go to System > External Systems > PingOne for Enterprise Settings, configure various PingOne for Enterprise integration settings and optionally enable and configure a built-in RADIUS server to integrate PingID with your VPN.

Configuring PingOne for Enterprise settings

To configure the PingOne for Enterprise settings, adjust the various integration settings.

Steps

1. Go to System > PingOne for Enterprise Settings.

2. Configure the PingOne for Enterprise integration settings. For more information, see the following table.

Field	Description
Enable Single Sign-On from PingOne to the PingFederate Administrative Console	Toggles the ability to sign on to the administrative console using the PingOne admin portal credentials.
Enable Monitoring of PingFederate from PingOne	Toggles the ability to monitor your PingFederate server (or servers in a clustered environment) from the PingOne admin portal
Rotate Key	Update the authentication key that PingFederate uses to communicate with PingOne for Enterprise. **Note** Periodic rotation can ensure optimal security of your environment. **Note** PingFederate also automatically rotates the signing certificate used by the managed service provider (SP) connection. Learn more in Managed SP connection to PingOne for Enterprise and signing certificate.
Launch PingOne Admin Portal	Use to access the PingOne admin portal.

Field	Description
Disconnect from PingOne	Use to disconnect PingFederate from your PingOne account. This is applicable if you have made changes that you should not propagate to your PingOne for Enterprise account. For instance, you have two PingFederate environments, testing and production. The production PingFederate server is configured with a managed SP connection to PingOne for Enterprise, but the PingFederate test server is not. You have just exported a configuration archive from the production server and imported it to the test server. As soon as the configuration archive is imported, the administrative console prompts you to decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise. In this example, you should disconnect the test server from PingOne for Enterprise so that nothing is uploaded to your PingOne for Enterprise account from the test server.

3. Save your configuration.

Configuring PingOne for Enterprise SSO settings

About this task

Use PingOne for Enterprise to enable single sign-on (SSO) and upload configuration changes to your account.

Steps

- To enable single sign-on (SSO) through PingOne for Enterprise, click Identity Repository Configuration.

> **Note**
>
> This is applicable if you have not yet completed the **PingOne SSO** configuration in the past, which would have created a managed service provider (SP) connection to PingOne for Enterprise.

- To upload configuration changes to your PingOne account, go to the PingOne for Enterprise settings window and select Update PingOne Identity Repository, then confirm your decision.

  This is applicable if you have made changes that you should propagate to your PingOne for Enterprise account.

  For example, you are about to set up a new SAML application on PingOne for Enterprise that requires a telephone number of the user. Because the current attribute contract in the managed SP connection does not include an attribute for telephone number, you extend the attribute with a new attribute, `PrimaryTelephone`. After the connection is saved, the administrative console prompts you to decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise. In this example, you should upload the new configuration to PingOne for Enterprise so that the new `PrimaryTelephone` attribute is made available when you set up the new SAML application in PingOne for Enterprise.

Enabling and configuring the built-in RADIUS server to integrate PingID with your VPN

About this task

Use the PingID VPN (RADIUS) configuration wizard to merge PingID with your own VPN.

Steps

1. Click PingID Configuration to open the PingID VPN (RADIUS) configuration wizard.

   This is applicable if you have not completed the PingID VPN (RADIUS) configuration in the past, which would have created a PingID provisioning connection, an instance of the PingID password credential validator (PCV), or both.

2. Use the wizard to configure the server.

Configuring SSO from the PingOne for Enterprise admin portal to the PingFederate administrative console

You can single sign-on (SSO) to the PingFederate administrative console from PingOne for Enterprise and configure authentication procedures as desired.

About this task

In PingFederate 10.1 and later, you can connect to PingOne for Enterprise after the initial PingFederate setup by going to System > External Systems > Connect to PingOne for Enterprise.

Additionally, you can continue to sign on to the administrative console through native or alternative console authentication using the direct sign on page. You can also disable the direct sign on page to enforce the policy that administrators must SSO to the administrative console from the PingOne admin portal.

Steps

- To SSO to the administrative console:

  1. Start a web browser.
  2. Browse to the URL https://<*pf_host*>:9999/pingfederate/app, where <*pf_host*> is the network address of your PingFederate server, either an IP address, a host name, or a fully qualified domain name reachable from your computer.

     Result:

     If the SSO option is enabled on the PingOne for Enterprise Settings window and you have signed on to the PingOne admin portal, the PingFederate administrative console is made available. If you are not signed on to the PingOne admin portal, you are prompted to enter your PingOne admin portal credentials. Upon verification, the PingFederate administrative console is made available.

- To sign on through native or alternative console authentication:

  1. Start a web browser.
  2. <*pf_host*>:9999/pingfederate/app?service=page/directLogin, where <*pf_host*> is the network address of your PingFederate server, either an IP address, a host name, or a fully qualified domain name reachable from your computer.

- To disable native and alternative console authentication:

  1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
  2. Change the `pf.console.authentication` property value to Browse to the URL https:// `none`.
  3. Save the change and then restart PingFederate.

  > **Note**
  >
  > In a clustered PingFederate environment, you only need to modify the `run.properties` file on the console node.

  Result:

  After restart, the direct login page is disabled. Administrators can only SSO to the PingFederate administrative console from the PingOne admin portal at https://*Browse to the URL<pf_host>*:9999/pingfederate/app.

  To re-enable native or alternative console authentication, update the `pf.console.authentication` property accordingly and then restart PingFederate.

Monitoring PingFederate from the PingOne for Enterprise admin portal

After connecting PingFederate to PingOne for Enterprise, you can monitor PingFederate from the PingOne admin portal.

About this task

The PingOne admin portal displays your PingFederate server (or servers if you have a clustered PingFederate environment) with basic information such as the node index number, the IP address, and the connection status with the date last seen. For each server, you can also collect additional information such as CPU load, Java virtual machine (JVM) memory information, and system memory information.

Steps

1. Go to System > External Systems > PingOne for Enterprise Settings.
2. Optional: If you do not want to monitor PingFederate using the PingOne admin portal, clear the Enable Monitoring of PingFederate From PingOne check box and click Save.

Updating the PingOne for Enterprise identity repository

You can upload configuration changes to your connected PingOne for Enterprise account's admin portal or disconnect from your PingOne for Enterprise account if needed.

About this task

After a managed service provider (SP) connection to PingOne for Enterprise admin portal is established, PingFederate monitors configuration changes that can impact the connection, such as updates to the base URL or imports of configuration archives that include managed SP connections to PingOne for Enterprise. When PingFederate detects such changes, the administrative console prompts you to decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise in a banner message.

Steps

1. Go to System > External Systems > PingOne for Enterprise settings.

   Choose from:

   - To upload configuration changes to your PingOne for Enterprise account, click Update PingOne Identity Repository.

     > **Note**
     >
     > This is applicable if you have made changes that you should propagate to your PingOne for Enterprise account.
     > For example, you are about to set up a new SAML application on PingOne for Enterprise that requires a telephone number of the user. Because the current attribute contract in the managed SP connection does not include an attribute for telephone number, you extend the attribute with a new attribute, `PrimaryTelephone` . After the connection is saved, the administrative console prompts you to decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise. In this example, you should upload the new configuration to PingOne for Enterprise so that the new `PrimaryTelephone` attribute is made available when you set up the new SAML application in PingOne for Enterprise.

   - To disconnect PingFederate from your PingOne for Enterprise account, click Disconnect from PingOne.

     > **Note**
     >
     > This is applicable if you have made changes that you should not propagate to your PingOne for Enterprise account.
     > For instance, you have two PingFederate environments, testing and production. The production PingFederate server is configured with a managed SP connection to PingOne for Enterprise, but the PingFederate test server is not. You have just exported a configuration archive from the production server and imported it to the test server. As soon as the configuration archive is imported, the administrative console prompts you to decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise. In this example, you should disconnect the test server from PingOne for Enterprise so that nothing is uploaded to your PingOne for Enterprise account from the test server.

Managing CAPTCHA and risk providers

PingFederate lets you add and configure risk provider instances. PingFederate includes plugins that support Google’s reCAPTCHA v2 Invisible and v3 services.

Before you begin

Before you can add an instance of a Google reCAPTCHA service provider, get a project ID and an API key pair from Google. The key pair consists of a site key and secret key. Learn more in Google’s reCAPTCHA Developer’s Guide.

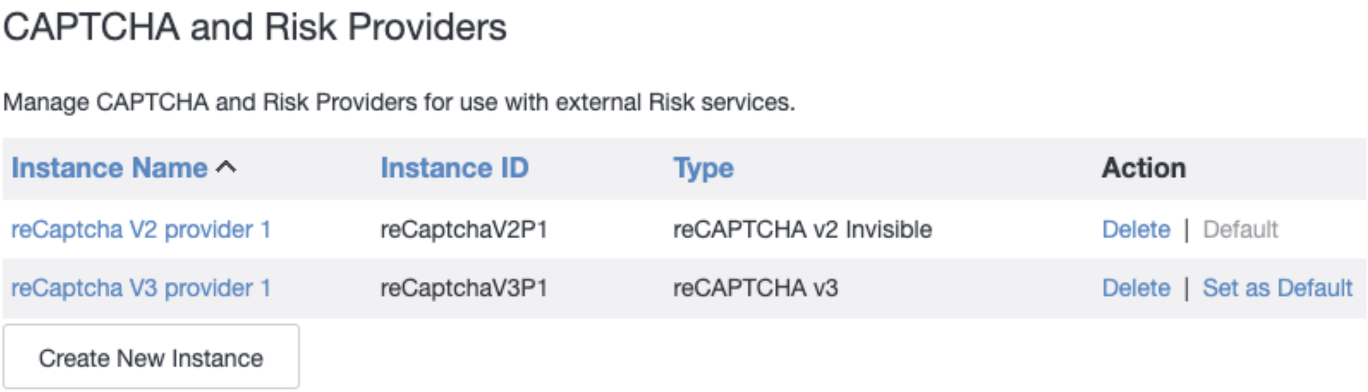
If you’re replacing the default JavaScript file to be loaded for reCAPTCHA, ensure the replacement file is in the `<pf_install>/pingfederate/server/default/conf/template/assets/scripts/captcha` directory.

About this task

The following procedure explains how to add and configure an instance of a reCAPTCHA service provider.

Steps

1. Go to System > External Systems > CAPTCHA and Risk Providers.



CAPTCHA and Risk Providers

Manage CAPTCHA and Risk Providers for use with external Risk services.

Instance Name ^	Instance ID	Type	Action
reCaptcha V2 provider 1	reCaptchaV2P1	reCAPTCHA v2 Invisible	Delete | Default
reCaptcha V3 provider 1	reCaptchaV3P1	reCAPTCHA v3	Delete | Set as Default

Create New Instance

2. Click Create New Instance. The Create Risk Provider Instance page opens.

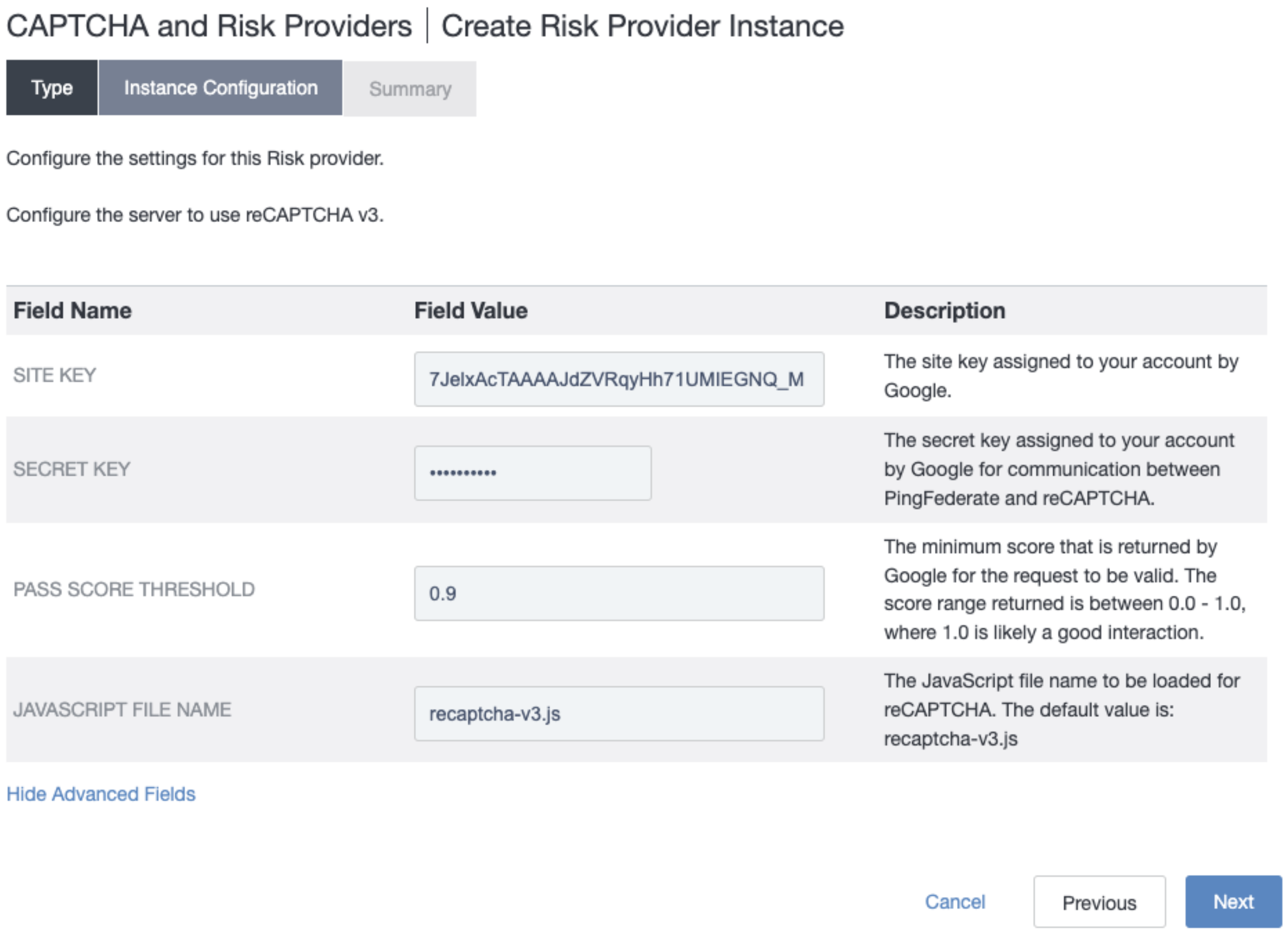
3. On the Type tab, enter a provider Instance Name and Instance ID. Select a type.

4. On the Instance Configuration tab:

    1. Enter the Site Key and Secret Key from Google.

    2. If you’re adding a reCAPTCHA v3 instance, enter the Pass Score Threshold.

    3. (Optional) To replace the default JavaScript file to be loaded for reCAPTCHA, click Show Advanced Fields and enter the new JavaScript File Name.



CAPTCHA and Risk Providers | Create Risk Provider Instance

Configure the settings for this Risk provider.

Configure the server to use reCAPTCHA v3.

Field Name	Field Value	Description
SITE KEY	7JelxAcTAAAAJdZVRqyHh71UMIEGNQ_M	The site key assigned to your account by Google.
SECRET KEY	••••••••••	The secret key assigned to your account by Google for communication between PingFederate and reCAPTCHA.
PASS SCORE THRESHOLD	0.9	The minimum score that is returned by Google for the request to be valid. The score range returned is between 0.0 - 1.0, where 1.0 is likely a good interaction.
JAVASCRIPT FILE NAME	recaptcha-v3.js	The JavaScript file name to be loaded for reCAPTCHA. The default value is: recaptcha-v3.js

5. On the Summary tab, review the configuration.

6. Click Save.

Managing SMS provider settings

To connect PingFederate to Twilio as an SMS provider through which PingFederate can send text message notifications for self-service password reset requests, enter the required information based on your Twilio account.

Steps

1. Go to System > External Systems > SMS Provider Settings.

2. Click Manage SMS Provider Settings.

3. On the SMS Provider Settings window, enter the required information.

Field	Description
Account SID	The account number assigned to your account by Twilio.

Field	Description
Auth Token	The password assigned to your account by Twilio. Used in conjunction with the account number to authenticate with Twilio when PingFederate makes outbound API calls for the purpose of sending text message notifications to the intended recipients.
From Number	The sender number in the text message notifications.

> **Tip**
>
> For additional information about each field or Twilio, see http://support.twilio.com.

4. Click Save.

Result

After you have saved them, these SMS provider settings apply to all services using text message notifications.

Managing notification publisher instances

Use PingFederate's functionality to create, edit, review, delete, or set as a default any notification publisher instance.

About this task

PingFederate delivers messages to administrators and end users based on notification publisher settings. Depending on your use cases, you can create one or more notification publisher instances. For example, you might select an SMTP notification publisher instance to deliver messages to your end users in an HTML Form Adapter instance, another SMTP notification publisher instance to deliver licensing messages to your fellow administrators, and an Amazon SNS notification publisher instance to deliver messages regarding SAML metadata updates.

Steps

- Go to System > External Systems > Notification Publishers.

  Choose from:

  - To configure a new instance, click Create New Instance.
  - To modify an existing instance, select it by its name under Instance Name.
  - To review the usage of an existing instance, click Check Usage under Action.
  - To remove an existing instance or to cancel the removal request, click Delete or Undelete under Action.
  - To make an instance the default notification publisher instance, click Set as Default under Action.

Defining a notification publisher instance

Define a notification publisher instance.

Steps

1. Go to System > External Systems > Notification Publishers.
2. Click Create New Instance.
3. On the Type tab, enter an instance name and an instance ID.
4. From the Type list, select the type of the notification publisher instance that you want to use.
5. Click Next.

Notification publisher instance configurations

You can configure a notification publisher instance based on its type.

This configuration varies depending on the type of the notification publisher.

For specific configuration steps, see the following topics..

- Configuring an Amazon SNS Notification Publisher instance
- Configuring an SMTP Notification Publisher instance

Configuring an Amazon SNS Notification Publisher instance

Create and configure an Amazon SNS notification publisher instance to help subscribers effectively direct their messages to their intended audiences.

About this task

When using an Amazon SNS notification publisher, PingFederate is the publisher, and the intended recipients are the subscribers. Topics are the destinations to which PingFederate publishes messages. When configuring an Amazon SNS notification publisher instance, you must specify an Amazon SNS topic.

For more information about Amazon SNS and topic management, see the AWS documentation on https://docs.aws.amazon.com/sns/latest/dg/welcome.html.

Steps

1. On the Instance Configuration tab, configure the notification publisher instance as follows.

Field	Description
SNS Topic ARN	The Amazon Resource Name (ARN) topic to which PingFederate publishes messages. Enter an ARN in the following format. `arn:aws:[service]:[region]:[accountid]:[resourceType/resourcePath]`

Field	Description
Max Payload Size	The maximum payload size in kilobytes. Enter a value between 1 and 8192. Click Show Advanced Fields to reveal this field. The default value is `256`.

2. Click Next to view the Summary tab, and then click Save.

Result

PingFederate categorizes notification messages into various event types. Each event type comes with a set of relevant information to help subscribers craft the final message for the intended audience.

Event types and variables

This lists the various event types and their respective keys (variables) used within the PingFederate Amazon SNS notification publisher instance configuration.

Message payload

As a publisher, PingFederate creates notification messages in JSON format and sends them to the configured topic. This JSON message body contains two top-level keys: `data` and `configuration`, as illustrated in the following snippet.

```
{
  "data": {
    "USERNAME": "jdoe",
    ...
  },
  "configuration": {
    "com.pingidentity.notification.config.locale": "en-US",
    ...
    "com.pingidentity.notification.config.event.type": "ADMIN_PASSWORD_CHANGED"
  }
}
```

For all events, PingFederate provides relevant information by including various *key*:*value* pairs in the message body, located inside the value of the `data` key.

The value of the `com.pingidentity.notification.config.event.type` key, located inside the value of the `configuration` key, indicates the event type. In this example, the event type is ADMIN_PASSWORD_CHANGED.

For end user-oriented events, the value of the `com.pingidentity.notification.config.locale` key, also located inside the value of the `configuration` key, indicates the locale of the end user who initiates the request.

Review the following sections for more information on event types and their respective keys, which are referred to as variables.

Events for administrators

Local administrative account management events

Event type	Variables
ADMIN_ACCOUNT_CHANGE_NOTIFICATION_OFF	- `USERNAME` represents the username of the local administrative account who has turned off the Notify Administrator of Account Changes option. - `RECEIVER` represents the email addresses of all the local administrative accounts configured with an email address. - `NOTIFY` represents the Notify Administrator of Account Change option on the Administrative Accounts window. - `CURRENT_USER_MESSAGE` represents the username of the administrator who initiated the change. ⓘ **Note** Unless otherwise noted, the rest of the variables in the **Administrative Accounts** section are either self-explanatory or identical to those mentioned here.
ADMIN_EMAIL_CHANGED	- `USERNAME` - `RECEIVER` - `DEPARTMENT` - `DESCRIPTION` - `PHONE_NUMBER` - `CURRENT_USER_MESSAGE` PingFederate sends two messages for this event type. Variables and their values remain the same, except for the `RECEIVER` value. They are intended to notify the end user at both the previous email address and the new email address.
ADMIN_PASSWORD_CHANGED	- `USERNAME` - `RECEIVER` - `DEPARTMENT` - `DESCRIPTION` - `PHONE_NUMBER` - `CURRENT_USER_MESSAGE`

Certificate, SAML metadata update, and licensing events

Event type	Variables
CERTIFICATE_EVENT_ACTIVATED and CERTIFICATE_EVENT_CREATED	• `SERIAL_NUMBER` • `SUBJECT_DN` • `EX_DATE` • `PENDING_CERT_SERIAL_NUM` • `PENDING_EX_DATE` • `ACTIVE_CONNECTIONS` represents the connections impacted by the creation of the pending certificate and the activation of it. • `ACTIVATION_DATE`
CERTIFICATE_EVENT_EXPIRED,CERTIFICATE_EVENT_FINAL_WARN, and CERTIFICATE_EVENT_INITIAL_WARN	• `SERIAL_NUMBER` • `SUBJECT_DN` • `EX_DATE` • `EX_TYPE` • `CONN_NAME` represents the connection impacted by any of the three certificate expiration events. • `DAYS_LEFT` • `ACTION`
SAML_METADATA_UPDATE_EVENT_ENTITY_ID_NOT_FOUND	• `ENTITY_ID` • `CONNECTION_NAME` • `METADATA_URL` • `METADATA_URL_NAME`
SAML_METADATA_UPDATE_EVENT_FAILED	• `METADATA_URL` • `METADATA_URL_NAME`
SAML_METADATA_UPDATE_EVENT_UPDATED	• `ENTITY_ID` • `CONNECTION_NAME` • `METADATA_URL` • `UPDATED` represents any updated connection settings. • `OUT_OF_SYNC` represents any out-of-sync connection settings.
SERVER_LICENSING_EVENT_WARNING, SERVER_LICENSING_EVENT_EXPIRED, and SERVER_LICENSING_EVENT_SHUTDOWN	• `EX_DATE` • `DAYS_LEFT`

Events for end users

Self-service password management, account recovery, and username recovery

Event type	Variables
ACCOUNT_UNLOCKED	• `USERNAME` represents the username of the end user where the request is made. • `RECEIVER` represents the email address of the end user where the request is made. • `ADAPTER_ID` represents the Instance ID of the invoking HTML Form Adapter instance. • `PCV_ID` represents the Instance ID of the Password Credential Validator (PCV) instance involved. ⓘ **Note** Unless otherwise noted, the rest of the variables in the HTML Form Adapter instances section are either self-explanatory or identical to those mentioned here.
PASSWORD_CHANGED	• `COMPANY_LOGO_EXISTS` • `globalKeyPrefix` • `ADAPTER_ID` • `PCV_ID` • `BASE_URL` • `params` • `locale` • `escape` • `templateMessages` • `FIRST_NAME` • `messageKeyPrefix`
PASSWORD_RESET	• `USERNAME` • `RECEIVER` • `ADAPTER_ID` • `PCV_ID` • `STATUS`
PASSWORD_RESET_FAILED	• `USERNAME` • `RECEIVER` • `ADAPTER_ID` • `PCV_ID`

Event type	Variables
PASSWORD_RESET_ONE_TIME_CODE and PASSWORD_RESET_ONE_TIME_LINK	• `USERNAME` • `RECEIVER` • `ADAPTER_ID` • `PCV_ID` • `CODE` represents the one-time code or hyperlink that the end user can use to reset the password associated with the account.
USERNAME_RECOVERY	• `USERNAME` • `RECEIVER` • `ADAPTER_ID` • `PCV_ID` • `DISPLAY_NAME`

Customer IAM email ownership verification

Event type	Variables
OWNERSHIP_VERIFICATION_ONE_TIME_LINK	• `USERNAME` represents the user name of the end user who should receive an email ownership verification request. • `RECEIVER` represents the email address to which the email ownership verification request is sent. • `CODE` represents the one-time hyperlink that the end user can use to verify the ownership of the email address associated with the account.

Configuring an SMTP Notification Publisher instance

Set up an instance of the SMTP Notification Publisher for PingFederate to notify administrators and end users about various events. You can configure multiple instances, each with different settings as needed.

Steps

1. On the Instance Configuration tab, provide the required information or update any default or previously configured setting values.

   For more information about each field, refer to the following table.

Field	Description
From Address (Required)	The email address that appears in the "From" header line in email messages generated by PingFederate. The address needs a valid format but does not need to be set up on your system.
Sender Name	The sender name displayed in email messages generated by PingFederate. Sender name cannot exceed 256 characters.
Email Server (Required)	The IP address or host name of your email server.
SMTP Port	The SMTP port on your email server. The default value is `25`.
Encryption Method	Select SSL/TLS to establish a secure connection to the email server at the SMTPS port. Select STARTTLS to establish an unencrypted connection to the email server at the SMTP port and initiate a secure channel afterward. Select None, the default value, to establish an unencrypted connection to the email server at the SMTP port.
SMTPS Port	The secure SMTP port on your email server. Not applicable unless SSL/TLS is the chosen encryption method. The default value is `465`.
Verify Hostname	Indicates whether to verify the host name of the email server matches the `Subject (CN)` or one of the `Subject Alternative Names` from the certificate. Not applicable unless either SSL/TLS or STARTTLS is the chosen encryption method. This check box is selected by default.
UTF-8 Message Header Support	Indicates whether the email server supports UTF-8 encoding in message headers; for example, in the recipient email address. Enable this option only if your email server supports this feature. With this option enabled, PingFederate supports UTF-8 characters in the sender and recipient email addresses. It does not support emojis in the domain portion of the email address.
Username	The username for the SMTP server.
Password	The password for the SMTP server.
Test Address	Enter an email address PingFederate should use to verify connectivity with the configured email server.
Click Show Advanced Fields to review the following settings. Modify as needed.	
Connection Timeout	The amount of time in seconds that PingFederate waits before it times out connecting to the SMTP server. The default value is `30`.

Field	Description
Enable SMTP Debugging Messages	Turns on detailed error messages for the PingFederate server log to help troubleshoot SMTP issues. ⓘ **Caution** This setting is disabled by default. When enabled, PingFederate logs email messages, which can contain sensitive information, to the server log. Consider enabling debug messages solely for troubleshooting purposes and disabling this option when debug messages are no longer required.

2. Click Next.

Next steps

You can modify email-notification template files to suit the particular branding requirements. For more information, see Customizable email notifications.

Finalizing actions for a notification publisher instance

Finish any required tasks presented to you on the Actions tab to complete configuration for your notification publisher instance.

About this task

> ⓘ **Note**
>
> Depending on the data store implementation, configuration requirements vary. If no action is required, the **Actions** tab is not shown.

Steps

- On the Actions tab, perform any required tasks, and click Next.

Reviewing a notification publisher instance configuration

On the Summary tab, review your configuration.

Steps

1. Perform the following actions as needed.

Action	How to accomplish it
Amend your configuration	Click the corresponding tab and follow the configuration workflow
Keep your changes	Click Done and continue with the configuration
Discard your changes	Click Cancel

Result

Once set up and saved, you can select this notification publisher instance in any component that is capable of triggering or handling events.

Secret managers

If you have a third-party secret management system (secret manager), you can configure instances of the PingFederate secret manager plugin, and use the instances to generate reference codes for your secret manager's contents.

In the secret manager, you can store the passwords and other credentials that PingFederate needs to access LDAP, JDBC, and REST API datastores. If you have a secret manager, you can integrate it with PingFederate and then configure one or more instances of the secret manager plugin. After using an instance to generate a reference code for a specific datastore credential in the secret manager, you can add the reference code to a PingFederate datastore plugin instance, allowing PingFederate to get the credential from your secret manager.

Storing datastore credentials in a secret manager is not only secure, it also lets you change the credentials in the secret manager without needing to change the configuration of the datastore plugins because the credential reference codes do not change. This is particularly useful when you rotate credentials.

You can integrate PingFederate with the CyberArk Credential Providerout-of-the-box. If you have a different secret manager, you can use the SecretManager interface in the PingFederate SDK to build a custom solution that connects PingFederate to your secret manager.

Integrating with the CyberArk Credential Provider

You can integrate PingFederate out-of-the-box with the CyberArk Credential Provider, an external secret management system (secret manager).

Before you begin

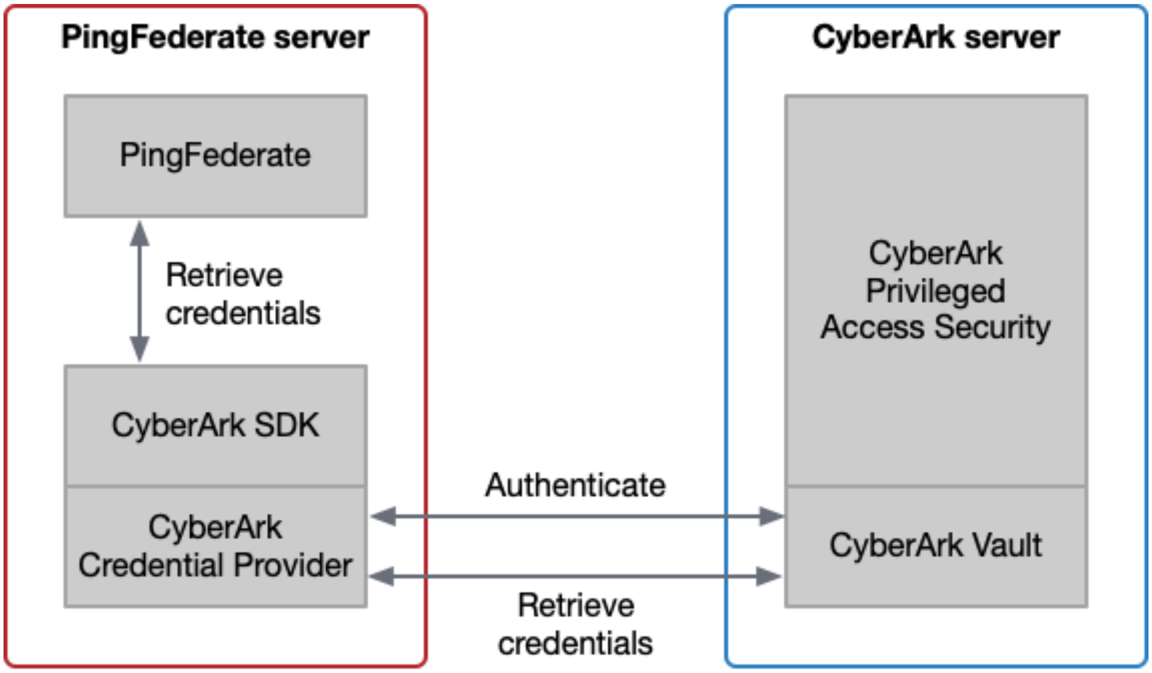
Install the Credential Provider and note the path to the CyberArk Java Application Password SDK file. The CyberArk website provides more information about the CyberArk Credential Provider and how to install it.

About this task

The Credential Provider supports the following authentication methods: allowed machines, OS user, path, and hash.

Note

Whenever you upgrade the CyberArk Credential Provider installation, replace the CyberArk Java Application Password SDK file with the latest version.



To integrate PingFederate with the CyberArk Credential Provider:

Steps

1. Copy the CyberArk Java Application Password SDK file to the `<pf_install>/pingfederate/server/default/deploy/` directory:
   - In Windows environments, copy the `JavaPasswordSDK.jar` file.
   - In Linux environments, copy the `javapasswordsdk.jar` file.

   Note

   You must install the SDK file on all nodes in your cluster. The active node needs the SDK file when performing the Validate action to check that it can retrieve the referenced secret from CyberArk. Passive nodes need to retrieve the secret at runtime.
2. Restart the PingFederate server.

Next steps

After integrating PingFederate with the CyberArk Credential Provider, you can:

- Configure instances of the secret manager in PingFederate.
- Configure authentication methods between the CyberArk Credential Provider and its CyberArk Vault. For more information, see CyberArk's authentication methods.

CyberArk's authentication methods

CyberArk administrators can configure one or more authentication methods between the CyberArk Credential Provider and its Vault.

The Credential Provider supports any combination of the following authentication methods:

- Hash
- OS user
- Allowed machines
- Path

For information about CyberArk's authentication methods, see Application authentication methods in the CyberArk documentation.

The following sections provide additional information specific to using the hash authentication method and OS user authentication method with PingFederate.

Hash authentication method

For information about using the hash authentication method, see Authenticate with a hash value in the CyberArk documentation.

> **Important**
>
> The hash changes when you perform a major or minor upgrade, or a maintenance update, of PingFederate. So you must regenerate the hash after an upgrade or update, otherwise PingFederate won't be able to retrieve credentials from CyberArk.

The following syntax and examples show how to use CyberArk's aimgetappinfo utility to generate a hash in Linux and Windows environments.

Linux syntax

```
aimgetappinfo GetHash -FilePath "<path to>/pf-core-plugins.jar"
```

Linux example command and its output

```
/opt/CARKaim/bin$ ./aimgetappinfo GetHash
   -FilePath "/home/imok/Downloads/pingfederate-11.0.0/pingfederate/server/default/lib/pf-core-plugins.jar"
<generated hash>
Command ended successfully
```

Windows syntax

```
AIMGetAppInfo GetHash /FilePath "<path to>\pf-core-plugins.jar"
```

Windows example command and its output

```
C:\Program Files (x86)\CyberArk\ApplicationPasswordProvider\Utils>AIMGetAppInfo GetHash
   /FilePath "C:\Users\Administrator\Downloads\pingfederate-11.0.0\pingfederate\server\default\lib\pf-core-
plugins.jar"
<generated hash>
Command ended successfully
```

OS user authentication method

For information about the OS user authentication method, see OS user authentication.

> **Note**
>
> In a Windows environment, if the PingFederate Windows service is installed or configured with Log On As: Local System, the CyberArk admin must enter `NT AUTHORITY\SYSTEM` as the OS user entry.

Configuring instances of the secret manager plugin for the CyberArk Credential Provider

To give PingFederate access to datastore credentials stored in your CyberArk Credential Provider, configure an instance of the CyberArk Credential Provider secret manager plugin.

Before you begin

Install the CyberArk Credential Provider and integrate it with PingFederate. For more information, see Integrating with the CyberArk Credential Provider.

> **Note**
>
> When configuring instances of the secret manager plugin, you need information about your secret manager's configuration. You also need information about the contents of your secret manager to generate reference codes for its contents.

About this task

To configure an instance of the secret manager plugin that provides access to the CyberArk Credential Provider:

Steps

1. In the PingFederate administrative console, go to System > External Systems > Secret Managers.

   Result:

   The Secret Managers window opens.

2. Click Create New Instance.

   Result:

   The Create Secret Manager Instance window opens.

3. Configure the Type tab settings:

   1. Enter an Instance Name and a unique Instance ID.

   2. In the Type menu, select CyberArk Credential Provider.

   3. Optional: To make this new secret manager instance the child of an existing instance, select the Parent Instance.

4. Configure the Instance Configuration tab according to the settings of your CyberArk Credential Provider.

   The App ID is the unique ID of the PingFederate application configured in the CyberArk Credential Provider.

5. Optional: On the Actions tab, verify that you can generate a valid reference code for a credential stored in the CyberArk Credential Provider:

   1. In the Generate section, enter each Parameter Value that PingFederate needs to retrieve a specific secret.

      The values depend on the name and location of the secret in the CyberArk Credential Provider. Optionally, you can specify in the reference code that PingFederate will also retrieve the username for the datastore account.

   2. Click Generate.

      Result:

      PingFederate generates and displays the secret's reference code. The code is composed of obfuscation prefix `OBF:MGR`, the plugin instance's ID, and the parameters you specify on this tab.

   3. Copy the reference code.

   4. In the Validate section, paste the code into the Secret Reference field.

   5. Click Validate.

      Result:

      PingFederate uses the reference code to request the secret from the CyberArk Credential Provider and then displays whether the request succeeded.

      > **Tip**
      >
      > To clear the fields and the generated reference code on the **Actions** tab, click **Reset**.

6. On the Summary tab, review the settings. Then, if needed, change the settings on the previous tabs.

7. Click Save.

   Result:

   The Secret Managers window opens, showing the new instance in the table.

Next steps

After configuring an instance of the secret manager plugin, use it to generate a reference code for a specific password in the CyberArk Credential Provider. Then you can add the reference code to the following places in PingFederate:

- An instance of a datastore plugin for an LDAP directory, JDBC database, or REST API. For more information, see Using passwords in secret managers to access datastores.
- The `ldap.properties` file, `oauth2.properties` file, and the `oidc.properties` file.

Using passwords in secret managers to access datastores

You can configure instances of PingFederate's datastore plugins to retrieve datastore account passwords that are stored in an external secret management system (secret manager).

Before you begin

Before performing this task, you must:

- Install the CyberArk Credential Manager or another secret manager
- Integrate the secret manager with PingFederate
- Add the datastore passwords to the secret manager
- Configure an instance of PingFederate's secret manager plugin to access the secret manager

About this task

Instead of storing passwords for LDAP directories, JDBC databases, and REST API datastores in PingFederate, you can securely store the passwords in a secret manager for maintaining passwords and other secrets. When PingFederate needs to access a datastore, it uses a reference code to request the password from the secret manager. However, before that can happen, you must generate a reference code for the datastore password and add it to the datastore plugin instance.

To generate a reference code for a datastore password and add it to a datastore plugin instance:

Steps

1. Use an instance of the secret manager plugin to generate a reference code for the datastore's password:
    1. In the PingFederate administrative console, go to System > External Systems > Secret Managers.

       Result:

       The Secret Managers window opens.
    2. Click the name of the secret manager plugin instance.

       Result:

       The Secret Manager window opens.
    3. Go to the Actions tab.
    4. In the Generate section, enter each Parameter Value that PingFederate needs to retrieve the datastore password.

The values depend on the name and location of the password in the CyberArk Credential Provider. Optionally, you can specify in the reference code that PingFederate will also retrieve the username for the datastore account.

5. Click Generate.

   Result:

   PingFederate generates and displays the password's reference code. The code is composed of obfuscation code `OBF:MGR`, the plugin instance's ID, and the parameters you specify on this tab.

6. Copy the reference code.

7. Optional: To verify that PingFederate can use the reference code to retrieve a password, paste the code into the Secret Reference field. Then click Validate.

   Result:

PingFederate requests the password from the CyberArk Credential Provider and then displays whether the request succeeded.

2. Add the password's reference code to the datastore plugin instance using one of the following methods, depending on whether the plugin is for an LDAP directory, JDBC database, or REST API datastore:

   Choose from:

   - For an LDAP directory, go to the plugin instance's LDAP Configuration tab, set Credential Storage to Secret Manager, and enter the Password Reference code that you generated above.
   - For a JDBC database, go to the plugin instance's Database Config tab, set Credential Storage to Secret Manager, and enter the Password Reference code that you generated above.
   - For a REST API datastore, go to the plugin instance's Configure Data Store Instance tab, and enter the Password Reference code that you generated above.

   If you configured the reference code with Retrieve Username enabled, PingFederate will ignore the username defined in the datastore plugin instance.

Troubleshooting

Basic troubleshooting tips are provided here to help you overcome common difficulties with PingFederate.

- Enabling debug messages and console logging
- Resolving startup issues
- Troubleshooting data store issues
- Resolving URL-related errors
- Resolving service-related errors
- Troubleshooting authentication policy issues
- Troubleshooting registration and profile management issues

- Troubleshooting runtime errors
- Troubleshooting OAuth transactions
- Other runtime issues
- Collecting support data

Help is also available from the Support Center.

Enabling console logging

You can edit the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file to enable console logging.

About this task

For troubleshooting purposes, you can enable console logging or verbose messages.

> **Important**
>
> When you no longer require console logging or verbose messages, turn them off. On Windows, never highlight the console output because it might slow or stop PingFederate from processing requests.

To enable console logging:

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.
2. Look for the `Set up the Root logger` section as shown in the following example.

```
...
<!-- ======================= -->
<!-- Set up the Root logger  -->
<!-- ======================= -->
...
<AsyncRoot level="INFO" includeLocation="false">
    <!-- <AppenderRef ref="CONSOLE" /> -->
    <AppenderRef ref="FILE" />
</AsyncRoot>
```

Then, update as shown in bold in the following snippet.

```
...
<!-- ======================= -->
<!-- Set up the Root logger  -->
<!-- ======================= -->
...
<AsyncRoot level="INFO" includeLocation="false">
     <AppenderRef ref="CONSOLE" />
    <AppenderRef ref="FILE" />
</AsyncRoot>
```

> **Important**
>
> When you no longer require console logging, comment out the `<AppenderRef ref="CONSOLE" />` entry for the `AsyncRoot` logger.

3. Save any changes made.

4. In a clustered PingFederate environment, repeat these steps on each applicable node.

Result

PingFederate activates the changes within half a minute. You don't need to restart PingFederate.

Related links

- Log4j 2 logging service and configuration

Resolving startup issues

Resolve PingFederate startup issues related to Java runtime and the license file installs.

PingFederate startup problems and solutions

Problem	Solution
PingFederate does not start.	Make sure that a supported Java runtime is installed. For more information, see Installing Java.
The server starts but indicates the license file is not found or invalid.	Ensure a current license is installed. For more information, see Reviewing license information.

Troubleshooting data store issues

Resolve PingFederate data store issues related to failing to establish a connection to a data store and directory.

Data store connection problems and solutions

Problem	Solution
When setting up the JDBC data store, a connection cannot be established.	Verify that the proper drivers and connectors have been installed. Also, verify the connection URL, user name, and password. If unsuccessful, contact your database administrator. For more information, see Configuring a JDBC connection.

Problem	Solution
Cannot connect to a Directory Service with the LDAP protocol.	Verify the connection URL, port, principal, and credentials. If unsuccessful, contact your system administrator. For more information, see Configuring an LDAP connection. If using LDAPS, ensure the LDAP server's SSL certificate is signed by a trusted certificate authority or is imported into PingFederate. For more information, see Manage trusted certificate authorities.

Resolving URL-related errors

Review and update the URL of the identity provider (IdP) initiated single sign-on to resolve the `404 Not Found` and `System Error` error messages.

If a user encounters a `404 Not Found` status or a `System Error` message, check the URL of the request.

ExampleExamples

404 Not Found

https://sso.idp.local/idp/startSSO.ping&PartnerSpId=sp1&TargetResource=https%3A%2F%2Fapp.sp1.local%2F causes a `404 Not Found` error, because the separator between the path of the URL and the first query parameter is incorrect. The correct separator is a question mark `?` and not an ampersand `&`.

System Error

https://sso.idp.local/idp/startSSO.ping?PartnerSpId=sp1?TargetResource=https%3A%2F%2Fapp.sp1.local%2F causes a `System Error` message, because the second query parameter separators are incorrect. The correct separator is an ampersand `&` and not a question mark `?`.

> **Tip**
>
> You must also use ampersands for all subsequent separators between additional query parameters in the URL.
> In addition, you must URL-encode query parameter values that contain restricted characters. For information about URL encoding, see, for example, HTML URL-encoding Reference.

For both sample issues, update the URL of the IdP-initiated single sign-on (SSO) to the following URL.

https://sso.idp.local/idp/startSSO.ping?PartnerSpId=sp1&TargetResource=https%3A%2F%2Fapp.sp1.local%2F

Resolving service-related errors

Resolve the `Unexpected System Error` message and `partner not active` status.

If a user encounters an `Unexpected System Error` message with a reference code, ask the user for the reference code and search for the value in the server log. The log message should help determine the root cause, which usually requires a configuration change.

If a user encounters a `partner not active` status, select `Active` in the Connection Status section and click Save on the Activation & Summary window for the connection.

ExampleExample

Unexpected System Error

When a PingFederate identity provider (IdP) server receives a SAML AuthnRequest message through the redirect binding, but such SAML profile is not selected in the applicable service provider (SP) connection, PingFederate replies with an `Unexpected System Error` response with a reference code and logs an error message similar to the following entry.

```
2015-12-03 15:43:52,936 ERROR [org.sourceid.servlet.ErrorServlet] Top level error (ref#kwlqbn):
javax.servlet.ServletException: org.sourceid.saml20.bindings.BindingException: Incoming binding
urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect is not enabled for (SP) ::: sp1
```

> **Tip**
>
> In this sample log message, `kwlqbn` is the reference code.

Solution

Update the applicable SP connection to allow the Redirect binding for inbound messages from the SP. This works if the redirect binding is one of the mutually-agreed SAML bindings that both parties use. Alternatively, the SP can send SAML AuthnRequest messages through an allowable SAML binding based on the configuration of your SP connection.

Troubleshooting authentication policy issues

Authentication policies, an optional configuration in PingFederate, help implement complex authentication requirements. Having a complex policy or multiple policies can result in unintended runtime behaviors. The `org.sourceid.util.log.PolicyTreeLogger` logger makes it easier to troubleshoot such issues.

About this task

Identify and resolve authentication policy issues.

Steps

1. Enable debug messages for the `org.sourceid.util.log.PolicyTreeLogger` class.

   > **Tip**
   >
   > If you have enabled debug messages for the `org.sourceid` logger, you have already enabled debug messages for the `org.sourceid.util.log.PolicyTreeLogger` class. Skip to step 3.

   1. Edit the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.
   2. Look for `Limit categories` inside the `Loggers` element, as shown in bold in the following example.

```
    ...
    <Loggers>

        <!-- ================ -->
        <!-- Limit categories -->
        <!-- ================ -->

        ...
         <!--
        <Logger name="org.sourceid.util.log.PolicyTreeLogger" level="DEBUG" />
         -->
        ...
    </Loggers>
    ...
    ```

    3. Uncomment the `org.sourceid.util.log.PolicyTreeLogger` logger, as shown in the following example. Note that the `<!--` and `-→` markers have been removed.

       Example:

       ```
       ...
       <Loggers>

           <!-- ================ -->
           <!-- Limit categories -->
           <!-- ================ -->

           ...

           <Logger name="org.sourceid.util.log.PolicyTreeLogger" level="DEBUG" />

           ...
       </Loggers>
       ...
       ```

   This change is activated within a minute. No restart of PingFederate is required.

2. Save any changes made.

   For a clustered PingFederate environment, repeat these steps on each applicable node.

3. Repeat the request that demonstrates the authentication policy issue.

   > **Tip**
   >
   > It is most useful to initiate this request in a browser without any cookies from prior sessions.

4. After you have replicated the issue, correlate server log messages using the PF cookie and tracking ID values. For more information, see Troubleshooting runtime errors.

   Look for `DEBUG` messages from the `org.sourceid.util.log.PolicyTreeLogger` class.

   Example:

For example, suppose the tracking ID value is `wXzQbS8MfHG40wpsQPiREIenJjc` for a given request. The following server log messages demonstrate the authentication flow.

```
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Selector |
generalClients | Yes
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
idFirst
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
idFirst
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
idFirst | Rule | Alpha
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
idFirst | Alpha
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
https://sso.alpha.local:8031
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Authn Policy Tree setting User ID from attribute
'subject' from Source type 'Adapter' and source ID 'idFirst'
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authn Source |
https://sso.alpha.local:8031 | Success
DEBUG [org.sourceid.util.log.PolicyTreeLogger] Policy 'General clients policy' | Authentication
Policy Contract | APC | Finished
```

> **Note**
>
> For readability, this sample ignores the time stamp and the tracking ID information. In other troubleshooting scenarios, such information can be valuable.

Log messages are interpreted as follows:

1. PingFederate finds an applicable policy named `General clients policy`. The first checkpoint is an OAuth Client Set Authentication Selector instance `generalClients`. PingFederate routes this request to the `Yes` policy path because the client that submits the authorization request matches one of the clients defined authentication selector instance.

2. PingFederate routes this request to an instance of the Identity First Adapter `idFirst` because that adapter instance is the next authentication source of the `Yes` policy path.

3. Based on the user's provided user identifier, PingFederate determines that the `Alpha` rule applies and routes this request to the `Alpha` policy path.

4. PingFederate routes this request to an identity provider (IdP) connection `https://sso.alpha.local:8031` because that IdP connection is the next authentication source of the `Alpha` policy path. PingFederate also populates the `subject` attribute in the AuthnRequest message with the user identifier obtained from the Identity First Adapter instance.

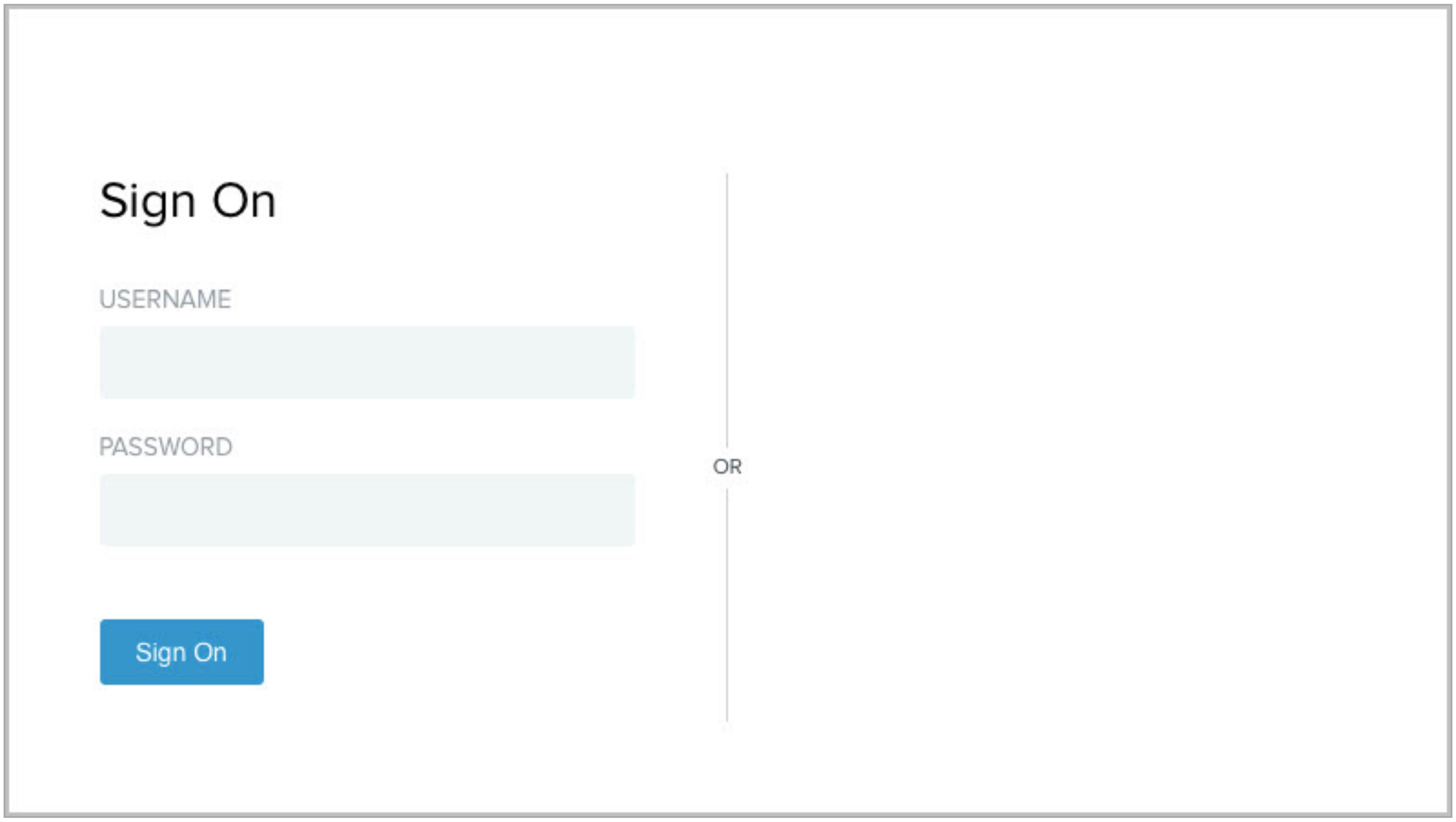
5. PingFederate receives a valid security token from the IdP `https://sso.alpha.local:8031`. PingFederate routes the request to the `Success` policy path, which ends with an authentication policy contract `APC` and concludes the authentication flow.

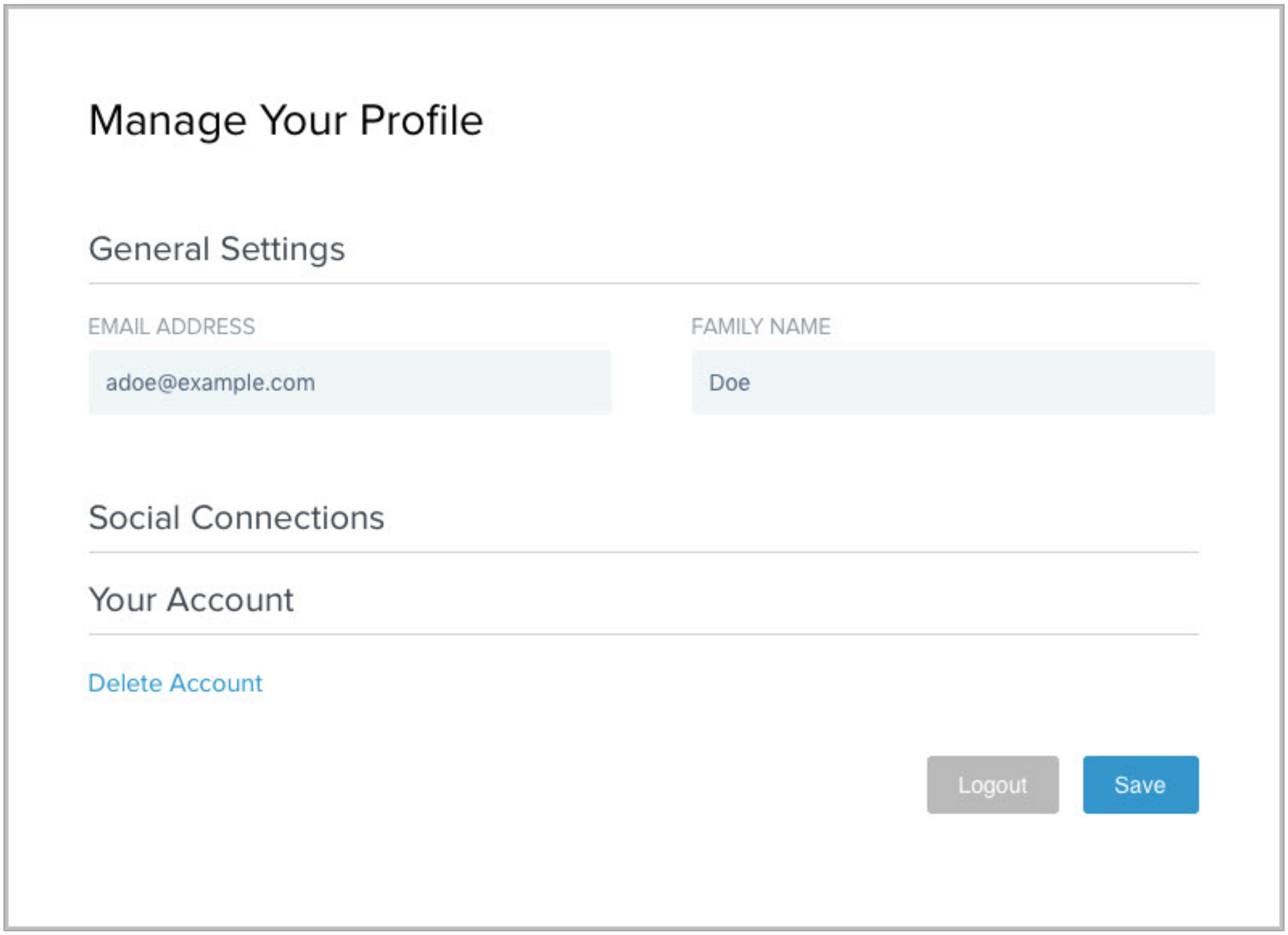
Troubleshooting registration and profile management issues

Certain browser extensions and their default settings can block a user's alternative authentication options configured to display on a third-party registration and profile management page. Review what browser a user currently employs and, if necessary, disable and fine-tune the extension settings of the browser.

About this task

Consider the following example. You have enabled third-party identity providers (IdP) to provide users with an alternative authentication option. Because you have enabled profile management, users can connect and disconnect their accounts with the third-party identity providers on the profile management page. However, some users are reporting that they do not see any such options on the sign-on page and the profile management page. The following screen captures illustrate what they see.





Manage Your Profile

General Settings

EMAIL ADDRESS: adoe@example.com

FAMILY NAME: Doe

Social Connections

Your Account

Delete Account

Browser extensions, particularly those designed to block ads or social network components, might block the user interface elements representing the third-party identity providers you enabled.

Steps

- Verify if users are using browser extensions that might have caused this issue. Disable or fine-tune the browser extension to resolve the issue.

Troubleshooting runtime errors

Users might encounter runtime errors when trying to connect to your partners. To troubleshoot these errors, investigate the user's activities, particularly the requests and the responses between the client and the PingFederate server, to determine the root cause.

Client side

Use built-in developer tools from the browsers to analyze HTTP traffic. Alternatively, you can use third-party tools, such as Fiddler and Charles.

Server side

Review server log messages to investigate what PingFederate has received from the client.On rare occasions, you can also use third-party tools, such as Wireshark, or tools from the operating systems, such as Tcpdump, to capture network traffic on the PingFederate server.

For instructions of third-party tools, see their documentation.

To correlate server log messages to user activities, you can use one of the following:

- The tracking ID, which can be configured to show in the user-facing error pages in PingFederate.
- The PF cookie value, which requires capturing the HTTP headers on the client side.
- Real-time monitoring of the server log, which works well if the issues can be replicated reliably and involves using tools from the operating systems or third-party vendors.

Activating tracking ID in templates

You can configure PingFederate to display the tracking ID in the user-facing error Velocity templates. When an error occurs, use the tracking ID to look for the related log messages.

About this task

You can find the Velocity template files in the `<pf_install>/pingfederate/server/default/conf/template` directory.

The Velocity variable is `$TrackingId` and is available in the following templates:

- `general.error.page.template.html`
- `generic.error.msg.page.template.html`
- `idp.slo.error.page.template.html`
- `idp.sso.error.page.template.html`
- `sourceid-wsfed-idp-exception-template.html`
- `sp.slo.error.page.template.html`
- `sp.sso.error.page.template.html`
- `state.not.found.error.page.template.html`

Steps

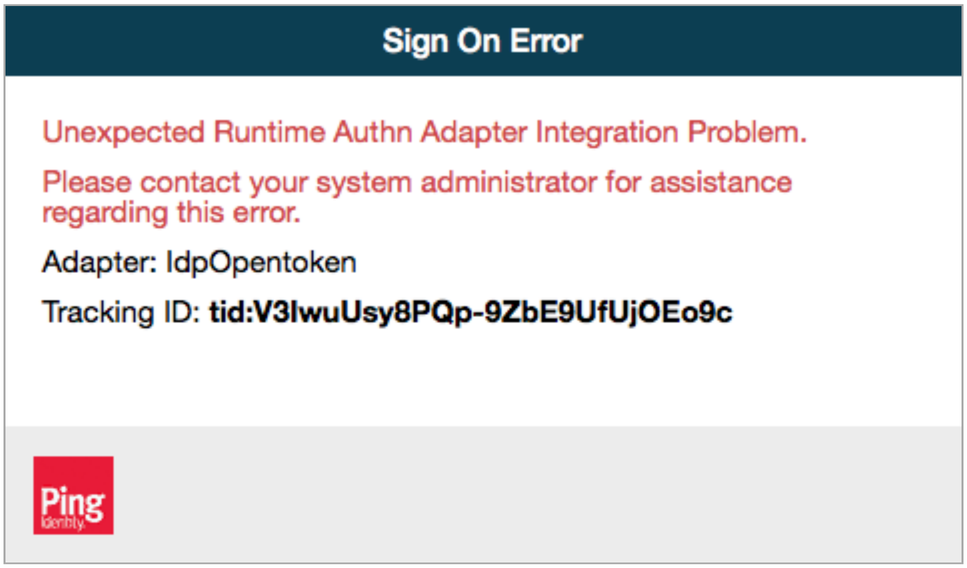
1. Open the applicable Velocity template file.
2. Search for the `$TrackingId` variable.
3. Follow the inline instructions to activate the variable.

> **Note**
>
> Template customization does not require a restart of PingFederate.

4. For a clustered PingFederate environment, repeat these steps on each engine node.

Result

The following screen capture demonstrates the user experience after the `$TrackingId` variable is activated and an error has occurred. In this example, `V3IwuUsy8PQp-9ZbE9UfUjOEo9c` is the tracking ID.



Sign On Error

Unexpected Runtime Authn Adapter Integration Problem.

Please contact your system administrator for assistance regarding this error.

Adapter: IdpOpentoken

Tracking ID: **tid:V3IwuUsy8PQp-9ZbE9UfUjOEo9c**

Correlating log messages by PF cookie

If you do not want to activate the tracking ID in the user-facing error templates, you can capture the HTTP traffic and use the PF cookie value to find related server log messages for a given request.

About this task

Search the HTTP traffic server log and use the PF cookie value to review server log messages.

Steps

1. Capture HTTP traffic and look for the PF cookie value.
2. Search for the PF cookie value in the server log.
3. Because all server log messages, except the contents of the inbound requests and the outbound responses, are prefixed with their respective tracking IDs, use the tracking ID to review log messages and payloads pertaining to this transaction.

Result

In general, the most useful log messages are the ones tagged with `WARN`, `ERROR`, or prefixed with `Caused by`.

Example

Suppose an error occurred and the associated the PF cookie value was `0axBwPGw5OBeHVXe1sgifB7iZR5Rz2VI4rhJwqUSIXV`. Based on the cookie value, you found the following log message.

```
2015-12-03 11:13:33,784 tid:V3IwuUsy8PQp-9ZbE9UfUjOEo9c DEBUG [org.sourceid.servlet.HttpServletRespProxy]
adding lazy cookie Cookie\{PF=OaxBwPGw5OBeHVXe1sgifB7iZR5Rz2VI4rhJwqUSIXV; path=/; maxAge=-1; domain=null}
replacing null
```

After reviewing the related log messages based on the tracking ID `V3IwuUsy8PQp-9ZbE9UfUjOEo9c`, you found the next few messages.

```
2015-12-03 12:36:21,176 tid:V3IwuUsy8PQp-9ZbE9UfUjOEo9c ERROR
[org.sourceid.saml20.profiles.idp.HandleAuthnRequest] Exception occurred during request processing
org.sourceid.websso.profiles.RequestProcessingException: Unexpected Runtime Authn Adapter Integration Problem.
```

```
…
```

```
Caused by: org.sourceid.saml20.adapter.AuthnAdapterException: Could not obtain attributes from the IdP
Authentication Service.
```

Based on these log messages, the remedy is to review and update the configuration of the applicable identity provider (IdP) adapter instance.

Correlating log messages by tracking ID

All server log messages, except the contents of the inbound requests and the outbound responses, are prefixed with their respective tracking IDs, which helps with locating related log messages and payloads for a given transaction for troubleshooting.

About this task

Review the server log messages using the `Tracking ID` value provided by the user.

Steps

1. Ask the user for the `Tracking ID` value in the error message.
2. Search for the tracking ID in the server log.
3. Review the log messages and payloads pertaining to the transaction that is associated with the tracking ID.

Result

In general, the most useful log messages are the ones tagged with `WARN`, `ERROR`, or prefixed with `Caused by`.

Example

Suppose an error occurred and the associated the tracking ID was `V3IwuUsy8PQp-9ZbE9UfUjOEo9c`. Based on the tracking ID, you found the following log message.

```
2015-12-03 11:13:33,784 tid:V3IwuUsy8PQp-9ZbE9UfUjOEo9c DEBUG [org.sourceid.servlet.HttpServletRespProxy]
adding lazy cookie Cookie\{PF=OaxBwPGw5OBeHVXe1sgifB7iZR5Rz2VI4rhJwqUSIXV; path=/; maxAge=-1; domain=null}
replacing null
```

After reviewing the related log messages, you found the next few messages.

```
2015-12-03 12:36:21,176 tid:V3IwuUsy8PQp-9ZbE9UfUjOEo9c ERROR
[org.sourceid.saml20.profiles.idp.HandleAuthnRequest] Exception occurred during request processing
org.sourceid.websso.profiles.RequestProcessingException: Unexpected Runtime Authn Adapter Integration Problem.
```

```
...
```

```
Caused by: org.sourceid.saml20.adapter.AuthnAdapterException: Could not obtain attributes from the IdP
Authentication Service.
```

Based on these log messages, the remedy is to review and update the configuration of the applicable identity provider (IdP) adapter instance.

Correlating PingFederate events with PingDirectory LDAP activities

When enabled on PingDirectory, you can correlate events in PingFederate with LDAP activities in PingDirectory by looking for the matching session and request tracking IDs in their logs.

PingFederate can receive many requests during a session. The session ID is consistent throughout a session, but the request ID is unique for every request. You can add the request ID to your log files and use it to search for specific events within a session.

PingFederate records runtime requests in its audit log and gives them tracking IDs. When PingFederate sends an LDAP call to PingDirectory, PingFederate also sends the request's tracking ID.

> **Important**
>
> For PingFederate to send the tracking ID to PingDirectory, the Intermediate Client Request Control (OID=1.3.6.1.4.1.30221.2.5.2) must be enabled in PingDirectory. Also, there cannot be any access control instructions that prevent the PingFederate service account accessing PingDirectory from using this OID.

PingDirectory records the tracking ID as a session ID or request ID value in its access log. In the log, the ID is a property of a via element.

For example, if you see the following via elements in the PingDirectory access log, you can match them with PingFederate events by looking for session ID `kkLivppizq1RvbaYBAuB1r9z-Y8'` and request ID `FhMl5Lz0KwsQphYUlUVHS4xkC5s` in the PingFederate audit log.

```
via="app='PingFederate' sessionID='tid:kkLivppizq1RvbaYBAuB1r9z-Y8'"
via="app='PingFederate' requestID='tid:FhMl5Lz0KwsQphYUlUVHS4xkC5s'"
```

> **Note**
>
> When a PingFederate endpoint receives a request, it records a request ID at the DEBUG level of the server log. When the **Request Header for Correlation ID** field in the **General Settings** window specifies a request header, if the request includes that header, and the header's value contains 1 to 50 alphanumeric characters and hyphens, then PingFederate uses that value for the request ID. Otherwise, PingFederate uses a unique value that it generates for the request ID. For more information, see General settings.

Steps

1. Open the `<pingfed-install>/pingfederate/server/default/conf/log4j2.xml` file.

2. Search for `fileName="${sys:pf.log.dir}/audit.log"`.

Note

Depending on which version of PingFederate you’re running, there will be multiple results. You’re looking for `SecurityAudit2File` .

3. In the `<pattern>` attribute, add the `%X{httpRequestId}` variable.

   Example

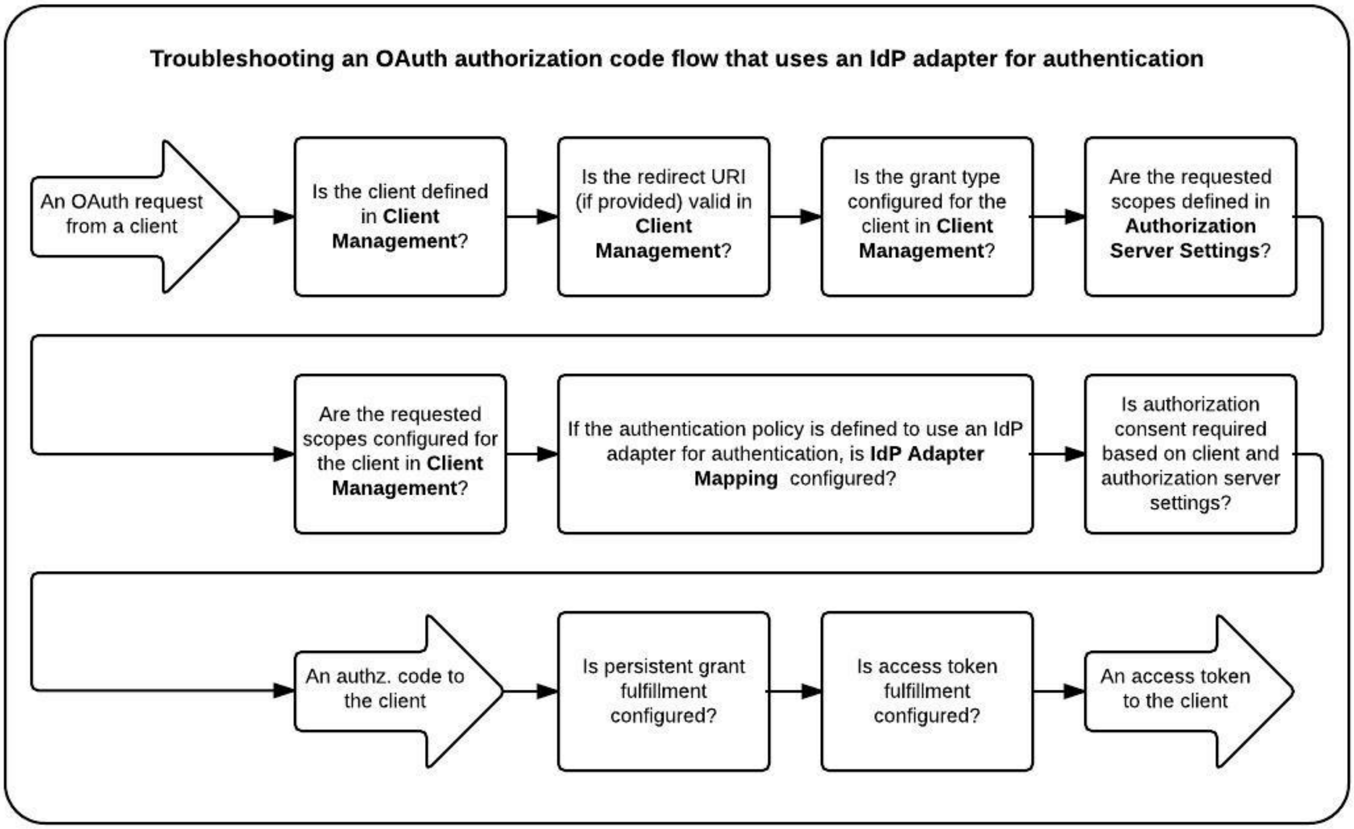
   ```
   <pattern>%d| %X{trackingid}| %X{httpRequestId}| %X{event}| %X{subject}| %X{ip} | %X{app}|
   %X{connectionid}| %X{protocol}| %X{host}| %X{role}| %X\status}| %X{adapterid}| %X{description}|
   %X{responsetime} %n</pattern>
   ```

4. Save and close the file.
5. If you’re running PingFederate in a clustered environment, repeat this process on each node.

Troubleshooting OAuth transactions

This troubleshooting guide walks through an OAuth request by inspecting the parameters provided by the client in different stages of an OAuth transaction. It then compares the parameter values against the corresponding settings defined in the PingFederate administrative console.

Troubleshooting OAuth use cases involves reviewing the OAuth requests and various OAuth settings.


Troubleshooting an OAuth authorization code flow that uses an IdP adapter for authentication
An OAuth request from a client
Is the client defined in Client Management?
Is the redirect URI (if provided) valid in Client Management?
Is the grant type configured for the client in Client Management?
Are the requested scopes defined in Authorization Server Settings?
Are the requested scopes configured for the client in Client Management?
If the authentication policy is defined to use an IdP adapter for authentication, is IdP Adapter Mapping configured?
Is authorization consent required based on client and authorization server settings?
An authz. code to the client
Is persistent grant fulfillment configured?
Is access token fulfillment configured?
An access token to the client

While the guide focuses on an OAuth authorization code use case, in which the end user authenticates through an identity provider adapter, it provides a general guidance for other OAuth use cases with or without OpenID Connect in terms of which part of the configuration comes into play and how a request might fail at which stage.

Related links

- OAuth configuration

Reviewing an OAuth request and various OAuth settings

To troubleshoot OAuth requests, inspect the parameters provided by the client in different stages of an OAuth transaction, and then compare the parameter values against the corresponding settings defined in the PingFederate administrative console.

About this task

A typical OAuth request looks like the following with the parameters that are submitted by a client. These are what you track as you go through the configuration during a troubleshooting task. Your requests could look very different depending on the specifics of your authorization server, resource server, and clients.

```
?client_id=client&response_type=code&redirect_uri=uri&scope=scope1 scope2
```

In this example request, the client is providing 4 parameters:

- client_id
- response_type
- redirect_uri
- scope

> **Note**
>
> There are other optional parameters that can be included in the request but are at this time not of interest to you in troubleshooting a typical request.

Steps

1. Review OAuth request in the server log.

   PingFederate logs requests and responses to the server log. The details vary based on the log level set in `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file. The following example shows a client making an Authorization Code grant type, as indicated by the `response_type` parameter of `code`.

```
2015-11-29 22:11:35,795 tid:aUBgyTMjPfuSFp5BYtsK6Cb2McM DEBUG
[org.sourceid.websso.servlet.ProtocolControllerServlet] ---REQUEST (GET)/as/authorization.oauth2
from 127.0.0.1:
    ---PARAMETERS---
    scope:
       list_users
    response_type:
       code
    client_id:
       pa_web_session_9
```

2. Check if the client is valid.

   1. Go to Applications > OAuth > Clients to open the Clients window.

   2. Look for the client by its Client ID.

      Result:

      If the client is not found, PingFederate denies the request, returns a 400 error to the client, and logs an `Unknown or invalid client_id` message to the server log, similar to the following code.

```
2015-11-29 22:11:35,812 tid:aUBgyTMjPfuSFp5BYtsK6Cb2McM DEBUG
[org.sourceid.oauth20.handlers.HandleAuthorizationRequest]
Normal exception being handled during OAuth request processing:
org.sourceid.oauth20.handlers.AuthorizationRequestException: Unknown or invalid client_id
```

3. Check if the `redirect_uri` parameter value is valid for the client.

   1. Go to Applications > OAuth > Clients to open the Clients window.

   2. Go to the Clients window.

   3. Select the applicable client.

   4. Compare the `redirect_uri` parameter value against the values defined in the Redirect URIs field.

      Result:

      If the request comes without a `redirect_uri` parameter and the Redirect URIs field contains multiple entries, PingFederate denies the request, returns a 400 error to the client, and logs an `Invalid redirect_uri` message to the server log, similar to the following code.

      ```
      2015-11-29 22:23:59,858 tid:aUBgyTMjPfuSFp5BYtsK6Cb2McM DEBUG
      [org.sourceid.oauth20.handlers.HandleAuthorizationRequest] Normal exception being handled
      during OAuth request processing: org.sourceid.oauth20.handlers.AuthorizationRequestException:
      Invalid redirect_uri
      ```

   If the request comes with a `redirect_uri` parameter value that does not match any Redirect URIs values defined for the client, PingFederate denies the request, returns a 400 error to the client, and logs an `Invalid redirect_uri` message to the server log.

4. Check if the `response_type` parameter value is authorized for the client.

    1. Go to Applications > OAuth > Clients to open the Clients window.

    2. Select the applicable client.

    3. In the Allowed Grant Types field, verify the `response_type` is selected.

        Result:

        If the `response_type` value is not one of the allowable grant types, PingFederate denies the request, returns a 403 error to the client, and logs an `unauthorized_client` message with an error description to the server log, similar to the following code.

```
2015-11-29 22:25:51,212 tid:aUBgyTMjPfuSFp5BYtsK6Cb2McM DEBUG
[org.sourceid.saml20.bindings.LoggingInterceptor] Transported Response. OutMessageContext:
    OutMessageContext
    entityId: pa_web_session_1 (null)
    virtualServerId: null
    Binding: oauth:authz
    params: {error=unauthorized_client, error_description=implicit grant not allowed for this
client}
    Endpoint: https://servapp.ext.den-ping.com/pa/oidc/
cb#error_description=implicit+grant+not+allowed+for+this+client&error=unauthorized_client
    SignaturePolicy: BINDING_DEFAULT
```

5. Check if the scopes requested with the `scope` parameter are defined for the authorization server.

    1. Go to System > OAuth Settings > Authorization Server Settings.

    2. In the Authorization Server Settings window, compare the scopes requested against the values defined in the Scope Value or the Scope Group Value fields.

        Result:

        If the scopes requested are not defined, PingFederate denies the request, returns a 403 error to the client, and logs an `invalid_scope` message with an error description to the server log, similar to the following code.

```
2015-11-29 22:24:52,588 tid:aUBgyTMjPfuSFp5BYtsK6Cb2McM DEBUG
[org.sourceid.saml20.bindings.LoggingInterceptor] Transported Response. OutMessageContext:
    OutMessageContext
    entityId: pa_web_session_1 (null)
    virtualServerId: null
    Binding: oauth:authz
    params: {error=invalid_scope, error_description=The requested scope(s) must be blank or a subset
of the provided scopes.}
    Endpoint: https://servapp.ext.den-ping.com/pa/oidc/cb?
error_description=The+requested+scope%28s%29+must+be+blank+or+a+subset+of+the+provided+scopes.&error=invalid_sc
    SignaturePolicy: BINDING_DEFAULT
```

6. Check if the scopes requested are valid for the client.

    1. Go to Applications > OAuth > Clients to open the Clients window.

   2. Select the applicable client.

   3. If the client is limited to specific scopes, as indicated by the selection of the Restrict Scopes check box, verify the scopes requested are valid for the client.

      Result:

   If the scopes requested are not valid for the client, PingFederate denies the request, returns a 403 error to the client, and logs an `invalid_scope` message with an error description to the server log.

7. Review the authentication process.

   Suppose this OAuth request uses an identity provider (IdP) adapter for authentication. Check the IdP adapter mapping and the runtime selection made by the user.

   1. Go to Authentication > OAuth > IdP Adapter Grant Mapping.

   2. Verify an entry exists for the IdP adapter involved.

      Result:

      If more than one option is available, authentication policies can be used to select an authentication source. If no authentication policy is defined or applicable, the user is prompted with a list of all available authentication sources. The user can also save the preferred authentication source for later in the form of a `pfidpaid` cookie.

   If a selection was made and the authentication source is not defined for OAuth, an error is returned to the user.

8. Upon successful authentication, PingFederate presents to the user an authorization consent page or a redirection to a trusted web application that is responsible to prompt the user for authorization unless a bypass option is configured. Review the authorization approval settings.

   1. Go to System > OAuth Settings > Authorization Server Settings.

   2. Review the Reuse Existing Persistent Access Grants for Grant Types setting.

      Result:

      If the grant type is selected, the authorization consent page is bypassed for the same client, the same user and same, or lesser, scope.

   3. Review the Consent User Interface setting.

      If PingFederate is configured to use an external consent user interface, verify that the associated settings are correctly configured and the web application is fulfilling its responsibilities.

   4. Go to Applications > OAuth > Clients to open the Clients window.

   5. Go to the Clients window.

   6. Select the applicable client.

   7. Review the Bypass Authorization Approval setting.

      Result:

   If the Bypass Authorization Approval check box is selected, the authorization consent page is bypassed as well.

9. Verify a mapping is configured.

When authorization is obtained, PingFederate maps attribute values from the authentication source into the persistent grants, the `USER_KEY`, `USER_NAME`, and extended attributes defined in the Authorization Server Settings window. This is the first stage of the two-stage OAuth attribute mapping process. In this example, because the user authenticates through an IdP adapter, check the IdP adapter mapping.

   1. Go to Authentication > OAuth > IdP Adapter Grant Mapping.
   2. Verify an entry exists for the IdP adapter involved and review its configuration.

10. Review the request and the settings related to access token management.

    Finally, PingFederate selects the applicable access token management (ATM) instance and fulfill the access token by mapping values from the persistent grants, the authentication source, or both. This is the second stage of the two-stage OAuth attribute mapping process.

    At runtime, the PingFederate OAuth authorization server uses the following rules to determine which ATM instances to use:

11. PingFederate limits the eligible ATM instances to those that are available in the context of the request. For most requests, these are instances that have an attribute mapping defined in the Access Token Mapping window. For OAuth Assertion Grant requests, it is the set of instances for which a mapping is defined in the IdP connection. If configured, the ACL can also limit which ATM instances are eligible.
12. If the request comes with an `access_token_manager_id` or `aud` parameter, PingFederate uses the information to determine the applicable ATM instance.
13. If the request does not come with either parameter, for OAuth clients supporting the OpenID Connect protocol by including the `openid` scope value, PingFederate uses the ATM instance specified by the OpenID Connect policy associated with the client. For resource server clients, you can optionally configure PingFederate to use any eligible ATM instances for the purpose of token validation.
14. If the request comes with neither of the two parameters nor the `openid` scope, PingFederate uses the default ATM instance of the client if configured, or the default ATM instance defined for the installation if eligible. For token validation requests, if resource server clients do not provide either the `access_token_manager_id` or `aud` parameter in their requests and the resource server clients have not been configured to validate against any eligible ATM instances, the same logic applies.

    1. Determine if the OAuth request is sent to the `/as/authorization.oauth2` authorization endpoint or the `/as/token.oauth2` token endpoint with an `access_token_manager_id` or `aud` parameter.
    2. Go to Applications > OAuth > Clients to open the Clients window.
    3. Select the applicable client.
    4. Verify if a default access token is selected from the Default Access Token Manager list.
    5. Go to Applications > OAuth > Access Token Mapping.
    6. Review the attribute mapping configuration for the authentication source, if such mapping exists, or the `Default` mapping.

Related links

- Configuring OAuth use cases

Other runtime issues

Possible runtime issues and their solutions related to unexpected certificate expiration and `CrossModule` and `Network` error messages.

Runtime issues and solutions

Problem	Solution
Certificates unexpectedly expire.	Verify that the server clocks are synchronized on both sides of the federation. You can configure PingFederate to notify administrators in advance of impending certificate expiration. For more information, see Runtime notifications.
Receive `CrossModule` or `Network` error messages when PingFederate is deployed with a supported hardware security module (HSM).	Verify network connections to the HSMs are active and running. Also ensure the HSMs have not been unintentionally shut down.

Collecting support data

When Ping Identity Support is helping an administrator troubleshoot a problem, it might ask the administrator to use the collect support data (CSD) tool to compile information about your PingFederate installation. The CSD tool is available through the `pingfederate/bin` directory.

About this task

The tool collects the following information by default:

- `pingfederate/bin`
- `pingfederate/log` (the most recent files of each type within a size limit)
- `pingfederate/server/conf` (configuration files)
- `pingfederate/server/data` (not key files)

The tool collects the following environment details:

- files present and their sizes
- certificate data
- version data
- JVM details
- and so on

The tool also collects the following system details, depending on the operating system:

- Crontab
- Ifconfig
- Netstat
- Uname
- and so on

If Ping Identity Support needs more information about the PingFederate installation than the default configuration provides, Support might ask the administrator to add a data collector to the tool by modifying its `csd_configuration.yaml` file.

The tool consists of the following files in the `pingfederate` directory:

- `bin/collect-support-data.bat`
- `bin/collect-support-data.sh`
- `bin/csd-1.0.jar`
- `server/default/conf/collect-support-data/csd_configuration.yaml`

> **Tip**
>
> You can use other parameters with the collect-support-data tool, including:
>
> **`--encrypt`**
>
> Encrypts the CSD output.
>
> **`--outputPath`**
>
> Specifies a directory for the CSD package.For additional information about these parameters and more, use the `--help` command.

Steps

1. Using your PingFederate administrator account and a terminal, navigate to the `pingfederate/bin` directory.
2. Use one of the following commands to run the CSD tool, depending on your operating system:

   Choose from:

   - On a Windows operating system, use `collect-support-data.bat` .
   - On a Unix-based operating system, use `./collect-support-data` .

   Result:

   As the tool collects data, it displays its progress and any errors. When it finishes collecting data, the tool places the data in a zip file in the `bin` directory. The file name format is `support-data-ping-${hostname}-r-${timestamp}.zip` .
3. Review any errors that the tool displayed during the process or added to the log file `collect-support-data.log` from the .zip file. If needed, resolve the errors and run the tool again.
4. Send the support data .zip file to Ping Identity Support according to the instructions from Support.

WS-Trust STS configuration

The PingFederate WS-Trust Security Token Service (STS) provides security-token validation and creation to extend single sign-on access to identity-enabled web services.

The section provides instructions for configuring the WS-Trust STS. For more information, see Web services standards.

> **Note**
>
> WS-Trust STS settings can also be configured through the PingFederate Administrative API platform. For more information, see Accessing the API interactive documentation.

Server settings

To use the PingFederate WS-Trust security token service (STS) for partner connections, enable and configure the WS-Trust protocol in your system server settings.

After you enable the protocol, you must identify the STS server with a unique federation identifier for both SAML 2.0 and SAML 1.1 tokens, unless these IDs are already established for the corresponding browser-based single sign-on protocols.

In addition, also within these server settings tasks, you have the option to require authentication globally for access to STS endpoints.

Enabling the WS-Trust protocol

To use the PingFederate WS-Trust security token service (STS) for partner connections, enable the WS-Trust protocol.

About this task

Using the administrative console, configure your server settings to enable the WS-Trust protocol.

Steps

1. Go to System > Server > Protocol Settings.
2. On the Federation Info tab, enter your SAML federation IDs unless these IDs are already established for the corresponding browser-based SSO protocols.

   > **Note**
   >
   > Identifiers are required for both SAML 2.0 and SAML 1.x to enable the STS to issue either type of token when requested. If you have not established a federation ID for either of these protocols or do not expect to use one or the other, enter a placeholder in any format and reconfigure later.

Configuring STS authentication

You can configure PingFederate to require that client applications provide credentials to access the STS.

About this task

Although it is an optional configuration, configuring security token service (STS) authentication is recommended for identity provider (IdP) configurations that use the Username Token Processor. For other token processors and token generators, trust in the identity of the client is conveyed within the token itself and verified as part of processing. You can still configure authentication requirements to add another layer of security by limiting access to only authenticated clients.

> **Note**
>
> You can configure STS authentication to either apply globally to all token formats and for all IdP and service provider (SP) partner connections, or token-to-token mappings, using more fine grained controls, at the connection level through issuance criteria.

Steps

1. Go to System > Server > Protocol Settings.
2. On the WS-Trust STS Settings tab, click Configure WS-Trust STS Authentication to open the WS-Trust STS Settings window.
3. On the Authentication Methods tab, select the Require HTTP Basic Authentication check box, the Require Mutual SSL/TLS Authentication check box, or both.

   If both the Require HTTP Basic Authentication check box and the Require Mutual SSL/TLS Authentication check box are selected, all clients must provide credentials for both mechanisms.

   > **Important**
   >
   > If you select the **Require Mutual SSL/TLS Authentication** check box, you must configure a secondary PingFederate HTTPS port `pf.secondary.https.port` in the `run.properties` file. For more information, see Configuring PingFederate properties.

4. If you select the Require HTTP Basic Authentication check box, manage user accounts on the HTTP Basic Authentication tab.
   1. Click Create User.
   2. In the HTTP Basic Authentication, enter a user name in the username field and a password in the password field.. Repeat to create additional user accounts for your client applications.
   3. Click Done.

   > **Note**
   >
   > On the **HTTP Basic Authentication** tab, you can also delete user accounts and update their passwords.

5. If you select the Require Mutual SSL/TLS Authentication check box, on the Mutual SSL Authentication tab, click Configure Mutual SSL Authentication.
   1. On the Authentication Options tab, you can select the Restrict Access by Subject DN check box and the Restrict Access by Issuer Certificate check box. Click Next.

      If both options are selected, the client certificate used for authentication to the STS endpoints must meet both sets of restrictions.
   2. If you selected the Restrict Access by Subject DN check box, enter one or more subject DNs on the Allowed Subject DNs tab.

> **Note**
>
> On the **Allowed Subject DNs** tab, you can edit or delete existing entries but you must keep at least one subject DN.

3. Click Next.. When finished, click Save.

4. If you selected the Restrict Access by Issuer Certificate check box, on the Allowed Issuer Certificates tab, from the Issuer Certificate list, select one or more client certificates.

5. Click Add.

   If you have not yet imported the client certificate, click Manage Certificates to do so.

> **Note**
>
> On the **Allowed Issuer Certificates** tab, you can remove existing entries but you must keep at least one issuer.

6. On the Summary tab, review your mutual SSL/TLS authentication settings. Click Done.

   Result:

   This will take you back to the WS-Trust STS Settings window.

6. When you finish configuring WS-Trust STS settings, on the Summary tab, review the configuration. To keep your changes, click Save.

Identity provider STS configuration

This section covers the identity provider (IdP) configuration for the PingFederate WS-Trust security token services (STS).

Managing token processors

The PingFederate Security Token Service (STS) uses token processors to validate incoming tokens and token requests.

About this task

You must configure at least one processor in order to set up an STS connection or a token-to-token mapping.

For more information about WS-Trust, see Web services standards.

PingFederate comes bundled with the following token processors:

- JWT Token Processor 1.2
- JWT Token Processor 2.0
- Kerberos Token Processor
- OAuth Bear Token Processor
- SAML 1.1 Token Processor

- SAML 2.0 Token Processor
- Username Token Processor

You can deploy additional token translators from Ping Identity website.

Steps

1. Go to Authentication > Token Exchange > Token Processors.
2. In the Token Processors window, choose from the following options.

Option	Description
Configure a new instance	Click Create New Instance
Modify an existing instance	Click the name of instance in the Instance Name column
View the usage of an existing instance	Click Check Usage in the Action column on the instance's row
Remove an existing instance	Click Delete in the Action column on the instance's row

> **Note**
>
> By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
> If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System** > **Server** > **General Settings**.

Selecting a token processor type

Begin creating a token processor instance by choosing the processor type.

Steps

1. Go to Authentication > Token Exchange > Token Processors.
2. Click Create New Instance. The Create Token Processor Instance window opens.
3. On the Type tab, enter a name in the Instance Name field and unique ID in the Instance ID field.
4. Select the token processor Type.
5. On the Parent Instance list, select a parent instance.

   This is useful when you are creating an instance that is similar to an existing instance. The child instance inherits the configuration of its parent. You can override settings during the rest of the setup by selecting the Override ... check box on the other tabs and making adjustments on them.
6. Click Save.

Next steps

After selecting the token processor type, go to the Instance Configuration tab to continue configuring the token processor instance. See Configuring a token processor instance.

Configuring a token processor instance

PingFederate configures multiple token processor instances. Select one that best fits your environment needs.

About this task

Depending on the selected token processor type selected on the Type tab, the Instance Configuration tab displays different parameters.

Steps

- To configure a specific token processor bundled with PingFederate, see one of the following topics:
  - Configuring a Username Token Processor instance
  - Configuring a Kerberos Token Processor instance
  - Configuring an OAuth Token Processor instance
  - Configuring a JWT Token Processor 1.2 instance
  - Configuring a JWT Token Processor 2.0 instance
  - Configuring a SAML Token Processor instance
- For information about add-on processors, see Integration overview.

Configuring a Username Token Processor instance

The integrated Username Token Processor accepts and validates username security tokens.

Steps

1. Go to Authentication > Token Exchange > Token Processors to open the Token Processors window.
2. Select on an existing token processor instance by clicking its name in the Instance Name section, or create a new instance by clicking Create New Instance.

   Result:

   This will open the Create Token Processor Instance window configuration.
3. On the Instance Configuration tab, configure the basics of this token processor instances.
   1. If you have not yet defined the desired Password Credential Validator instance, click Manage Password Credential Validators to do so.

2. Click Add a new row to 'Credential Validators' to select a credential-authentication mechanism instance for this adapter instance.

3. From the Password Credential Validator Instance list, select a Password Credential Validator instance. Click Update.Add as many validators as necessary. Use the up and down arrows to adjust the order in which you want PingFederate to attempt credential authentication. If the first mechanism fails to validate the credentials, PingFederate moves sequentially through the list until credential validation succeeds. If none of the Password Credential Validator instances can authenticate the user’s credentials, and the challenge retries maximum has been reached, the process fails.

   > **Note**
   >
   > If usernames overlap across multiple Password Credential Validator instances, this failover setup could lock out those accounts in their source locations.

4. In the Field Value section, enter a value in the Authentication Attempts.

When the number of login failures reaches this threshold, the user is locked out for a period of time. The default value is `3`.

Configuring a Kerberos Token Processor instance

The integrated Kerberos Token Processor accepts and validates Kerberos tokens through a configured Kerberos realm.

About this task

It supports authentication mechanism assurance from Active Directory (AD) domain service, making it possible to restrict access to users authenticating through specific mechanisms. For more information, see Authentication mechanism assurance.

Steps

1. Go to Authentication > Token Exchange > Token Processors.

2. On the Instance Configuration tab, select the applicable domain from the Domain/Realm Name list.

   An AD domain or a Kerberos realm must be configured for use with the Kerberos Token Processor. If the domain you want does not appear, click Manage Active Directory Domains/Kerberos Realms to add it. For more information, see Active Directory and Kerberos.

   > **Note**
   >
   > Kerberos tickets can be accepted from domains other than the domain configured in the token processor if there is a transient, two-way trust. This trust exists by default when domains are joined within a single server forest. For more information, see Multiple-domain support.

Configuring an OAuth Token Processor instance

The PingFederate STS provides validation for OAuth 2.0 bearer tokens. To use the OAuth Token Processor, you must first configure an Access Token Management (ATM) instance.

About this task

For more information about PingFederate OAuth authorization server and access token management, see About OAuth and Access token management.

Steps

1. Go to Authentication > Token Exchange > Token Processors.
2. On the Instance Configuration tab, configure the basics of the token processor instance.
    1. In the Access Token Manager row, from the Field Value list, select an ATM instance.

       If the desired ATM instance is not shown, click Manage Access Token Manager.

       Result:

       The token processor instance uses the selected ATM instance to validate the OAuth bearer access tokens.
    2. Optional: Select the Scope Value as Single String check box.

       Result:

       If selected, the scope value is returned as a single space-delimited set of string value. If it is not selected, scope values are returned as a multivalued attribute.

Configuring a JWT Token Processor 1.2 instance

The PingFederate security token services (STS) provides validation for JSON web tokens (JWTs).

About this task

When configuring a JWT Token Processor instance to validate incoming JWTs, the system relies on a JSON Web Key Set (JWKS) to verify the token's signature. For the validation to succeed, specific attributes in the JWK must match corresponding information in the JWT header.

You can find more information about each attribute in the following table:

Required JWK Attributes for JWT Validation

Attribute	Description
`kid`	The `kid` (key ID) parameter matches a specific key.

Attribute	Description
`use`	The `use` (public key use) parameter identifies the intended use of the public key. `use` indicates a public key is used for verifying the data signature. **Important** The parameter value must be `sig`.
`alg`	The `alg` (algorithm) parameter must match the `kty` (key type parameter), which is the cryptographic algorithm family used with the key.

Learn more in JSON Web Key (JWK).

Steps

To configure the JWT Token Processor Instance:

1. Go to Authentication > Token Exchange > Token Processors.
2. On the Instance Configuration tab, enter the required information.

   See the following table for information about each field.

JWT Token Processor instance field names and descriptions

Field	Description
JWKS Endpoint URI	The URI of the JWKS endpoint. A set of JSON Web Keys (JWK) are downloaded from this endpoint and used for JWT signature verification.
Issuer	A unique identifier for the issuer of the JWT.
Expiry Tolerance	The amount of time, in seconds, to allow for clock skew between servers. Valid range is 0 to 3600.

Configuring a JWT Token Processor 2.0 instance

The PingFederate Security Token Service (STS) provides validation for any JSON Web Token (JWT).

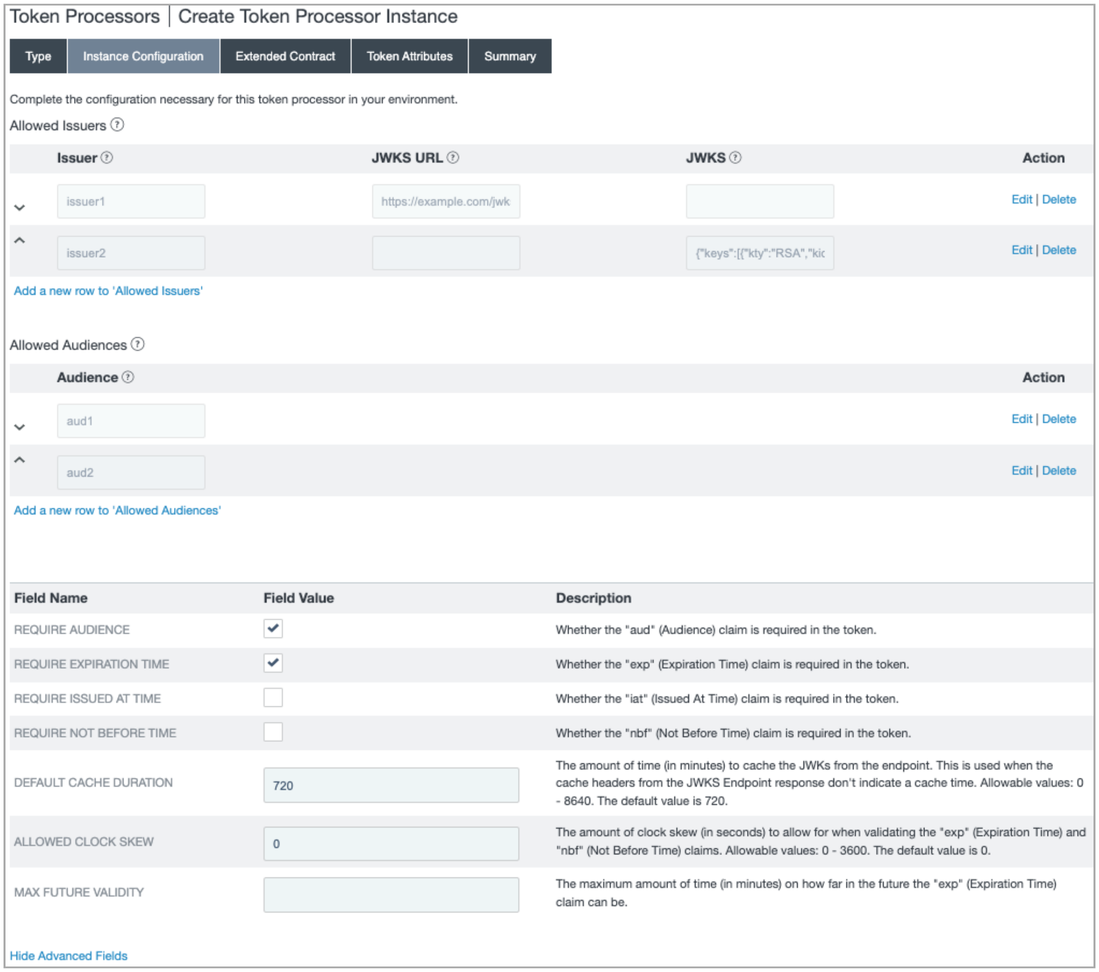
Before you begin

Use the Type tab on the Create Token Processor Instance window to begin configuring a JWT token processor 2.0 instance. See Selecting a token processor type.

About this task

The following procedure describes how to use the Instance Configuration tab on the Create Token Processor Instance window to continue configuring a JWT token processor 2.0 instance.

This feature supports the OAuth 2.0 Token Exchange and WS Trust specifications. JWT token processor 2.0 offers more functions than does JWT token processor 1.2.



Complete the configuration necessary for this token processor in your environment.

Allowed Issuers

Issuer	JWKS URL	JWKS	Action
issuer1	https://example.com/jwk		Edit | Delete
issuer2		{"keys":[{"kty":"RSA","kic	Edit | Delete

Add a new row to 'Allowed Issuers'

Allowed Audiences

Audience	Action
aud1	Edit | Delete
aud2	Edit | Delete

Add a new row to 'Allowed Audiences'

Field Name	Field Value	Description
REQUIRE AUDIENCE	☑	Whether the "aud" (Audience) claim is required in the token.
REQUIRE EXPIRATION TIME	☑	Whether the "exp" (Expiration Time) claim is required in the token.
REQUIRE ISSUED AT TIME	☐	Whether the "iat" (Issued At Time) claim is required in the token.
REQUIRE NOT BEFORE TIME	☐	Whether the "nbf" (Not Before Time) claim is required in the token.
DEFAULT CACHE DURATION	720	The amount of time (in minutes) to cache the JWKs from the endpoint. This is used when the cache headers from the JWKS Endpoint response don't indicate a cache time. Allowable values: 0 - 8640. The default value is 720.
ALLOWED CLOCK SKEW	0	The amount of clock skew (in seconds) to allow for when validating the "exp" (Expiration Time) and "nbf" (Not Before Time) claims. Allowable values: 0 - 3600. The default value is 0.
MAX FUTURE VALIDITY		The maximum amount of time (in minutes) on how far in the future the "exp" (Expiration Time) claim can be.

Hide Advanced Fields

Steps

1. On the Create Token Processor Instance window, go to the Instance Configuration tab.
2. Specify one or more Allowed Issuers and a JWKS or JWKS URL for each allowed issuer.

   PingFederate uses the JWKS or JWKS URL to get the validation keys for the issuer.
3. Specify one or more Allowed Audiences.

   This setting is optional unless you select the Require Audience check box.
4. Specify which of the following token claims are required:
   - Audience (`aud`)
   - Expiration time (`exp`)
   - Issued at time (`iat`)
   - Not before time (`nbf`)

By default, the `aud` and `exp` claims are required, and the `iat` and `nbf` claims are not required.

5. Optional: Click Show Advanced Fields and change the default value for any of the following settings:
   - Default Cache Configuration, which sets the number of minutes to cache the JWKS

     > **Note**
     >
     > This feature affects JWKS caching only when you specify a **JWKS URL** for an **Allowed Issuer** and the JWKS URL response doesn't indicate a cache time. This feature doesn't apply when you specify a **JWKS** for an allowed issuer.

   - Allowed Clock Skew for `exp` and `nbf` claims
   - Max Future Validity, which limits the lifetime of the token
6. Click Save.

Next steps

After selecting the token processor type, go to the Extended Contract tab to continue configuring the token processor instance. See Extending a token processor contract.

Configuring a SAML Token Processor instance

The integrated SAML (1.1 or 2.0) Token Processor accepts and validates SAML (1.1 or 2.0) security tokens. The PingFederate security token service (STS) validates digital signatures using all trusted certificate authorities (CAs) imported into PingFederate.

> **Important**
>
> The `Signature` element must include `KeyInfo` for signature verification to complete successfully.

About this task

On the Instance Configuration tab, configure a SAML Token Processor instance.

You can restrict the signature verification process by subject distinguished names (DN), issuers, or both to limit the token requests accepted for this token processor instance.

You must indicate a unique identifier for the PingFederate STS. Token processor instances reject SAML tokens that do not contain the identifier in the `audience` element.

Steps

- Go to Authentication > Token Exchange > Token Processors.
- On the Instance Configuration tab, configure the basics of the token processor instance.
  1. In the field value field of the Audience row, enter the URI that uniquely identifies your federation gateway for this SAML protocol.

This is the federation ID for the STS for either SAML 1.1 or SAML 2.0 tokens, depending on which processor you are configuring.

2. (Optional) Click Add a new row to 'Valid Certificate Issuer DNs' to enter one or more issuers.

> **Important**
>
> If issuer DNs are specified, only those issuers are considered valid for verifying incoming digital signatures. Otherwise, all trusted certificate authorities (CAs) are used to verify signatures.

3. (Optional) Click Add a new row to 'Valid Certificate Subject DNs' to enter one or more subject DNs.

> **Important**
>
> If subject DNs are specified, only those subject DNs are considered valid for verifying incoming digital signatures. Otherwise, all trusted certificate authorities (CAs) are used to verify signatures.
> If you specify both issuer DNs and subject DNs, then the certificate used to validate signatures must match an entry in both lists.
> If you provide no issuer DN and subject DN, then all certificates are treated as valid for purposes of verification.

Extending a token processor contract

Token processors allow administrators to add to a built-in list of user attributes that the processor returns from an incoming token.

Before you begin

Use the Instance Configuration tab on the Create Token Processor Instance window to configure the token processor instance. See Configuring a token processor instance.

About this task

> **Note**
>
> The **Extended Contract** tab shows a different list of attributes under **Core Contract**, depending on the token processor selected.

Steps

1. Go to Authentication > Token Exchange > Token Processors.
2. On the Extended Contract tab, in the Extend the Contract field, enter the name of the desired attribute. Click Add.

> **Important**
>
> For the OAuth Bearer Token Processor, added attributes must also be among those configured with the associated access token management instance.

Repeat these steps to add additional attributes.

Setting attribute masking

Mask attribute values that PingFederate logs from a processor instance at runtime.

Steps

1. Go to Authentication > Token Exchange > Token Processors.
2. On the Token Attributes tab, in the Mask Log Values section, select the check box for the attribute whose value you want to mask in logs.

   > **Note**
   >
   > If OGNL expressions might be used to map derived values into outgoing tokens and you want those values masked in logs as well, select the **Mask All OGNL-expression Generated Log Values** check box under the list of attributes.

Reviewing the token processor configuration

Review your token processor instance summary information.

About this task

On the Summary tab, review, amend, save, or discard the token processor configuration.

Steps

- To keep your changes, click Save.
- To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
- To discard your changes, click Cancel.

Managing STS request parameters

Configure PingFederate to act as a WS-Trust security token service (STS) by defining sets of Request for Security Token (RST) metadata parameters that can be used for mapping attributes into outgoing SAML security tokens.

About this task

After these request contracts are defined, you can make them available when configuring WS-Trust STS settings in the SP connections. For more information, see Selecting a request contract.

To manage your request contracts:

Steps

- Go to Authentication > Token Exchange > STS Request Parameters

  Choose from:

  - To configure a new set of request parameters, click Add New Request Contract.

 - To modify an existing request contract, select it by its name in the Contract Name column.

- To review the usage of an existing request contract, in the Action section, click Check Usage.
- To remove an existing request contract or to cancel the removal request, in the Action section, click Delete or Undelete.

Creating a request contract

Specify one or more parameters that will be included in request-security-tokens (RST) applicable to connection partners. You can make request contracts available for token-attribute mapping during partner-connection configuration.

About this task

Identify the contract and define parameters that will be available in token requests associated with this contract for partner connections.

Steps

1. Go to Authentication > Token Exchange > STS Request Parameters.

   Result:

   This will open the STS Request Parameters window configuration.

2. Click Add New Request Contract.

   Result:

   This will open the Request Contract configuration window.

3. Enter the contract name in the Contract Name field and the contract ID in the Contract ID field.

   > **Note**
   >
   > In the **Request Contract** configuration window, after you click **Done**, the **Contract Name** and **Contract ID** cannot be modified.

4. In the Parameter Name field, enter the parameter that will be included in the RSTs. Click Add.

   You must add at least one parameter. Repeat this step to add more parameters.

   > **Note**
   >
   > After the request is saved, you can add, modify, or remove parameters. You must keep at least one parameter.

5. Click Done.

Configuring SP connections for STS

Configure a security token service (STS) connection to a service provider (SP) partner either in conjunction with browser-based single sign-on (SSO) or independently.

Steps

1. Go to System > Server > Protocol Settings.
2. On the WS-Trust STS tab, configure an STS connection.

Configuring protocol settings for IdP STS

Specify the WS-Trust protocol details for web service clients related to this connection.

Steps

1. Go to Applications > Integration > SP Connections.
2. Click on an existing connection in the SP Connection column, or click Create Connection to configure a new SP connection to open the SP Connection configuration window.
3. On the WS-Trust STS tab, click Configure WS-Trust STS to open the WS-Trust STS configuration window.

   > **Note**
   >
   > The **WS-Trust STS** tab is only available after you enable the **WS-Trust** role on the**Connection Type** tab. For more information, see Configuring SP Connections for STS.

4. On the Protocol Settings tab, enter a URL for your partner’s web service in the Partner Service Identifier field. Click Add.

   This identifier compares to the `AppliesTo` element in the Requests for Security Token (RST) messages and can be either a complete URL or a base URL for matching variable ports or paths.

   Repeat this step to add additional identifiers.

5. Select any of the following WS-Trust protocol setting options that are applicable to your use case.

Option	Description
OAuth Assertion Profiles	When selected, four additional token-type requests become available based on these OAuth grant types: ◦ JWT Bearer Token grant type ◦ OAuth Access Token via JWT Bearer Token grant type ◦ SAML 2.0 Bearer Assertion grant type ◦ OAuth Access Token via SAML 2.0 Bearer Assertion grant type See STS OAuth integration for more information on the use of these token-type requests.

Option	Description
Default Token Type	The default token type when a web service client (WSC) does not specify in the token request which token type the STS should issue. The choices are: ◦ SAML 2.0 ◦ SAML 1.1 ◦ SAML 1.1 for Office 365 The default token type does not need to match the protocol selected for the browser-based SSO, if enabled, and does not apply to OAuth assertion profiles because those RST messages must contain the requested token type.
Generate Key for SAML Holder of Key Subject Confirmation Method	When selected, the STS generates a symmetric key to be used in conjunction with the "Holder of Key" (HoK) designation for the assertion’s Subject Confirmation Method. For information about HoK assertions, see Web Services Security SAML Token Profile. This option does not apply to OAuth assertion profiles.
Encrypt SAML 2.0 Assertion	When selected, the STS encrypts the SAML 2.0 assertion. Applicable only to SAML 2.0 security token. This option does not apply to OAuth assertion profiles.

6. On the Protocol Settings tab, customize SAML messages and assertions for WS-Trust connections. Message customizations are OGNL expressions that allow you to customize the security token sent from PingFederate to the service provider (SP).

    1. Click Show Advanced Customizations.

    2. From the Message Type list, select a type option and enter an expression. The message type is used to override the message type returned from the OGNL expression.

        The following tables describe the relationship between message type and available variables, and the corresponding class or interface information in Java.

SP connections SAML 2.0 message types and expressions

Message types	Available variables and classes/interfaces in Javadoc
AssertionType	#AssertionType org.sourceid.saml20.xmlbinding.assertion.AssertionType #AssertionTypes org.sourceid.saml20.xmlbinding.assertion.AssertionType[] #Attributes org.sourceid.util.log.AttributeMap

Message types	Available variables and classes/interfaces in Javadoc
ResponseDocument	#ResponseDocument For a connection with WS-Trust v1.3, #ResponseDocument will be of type org.oasisOpen.docs.wsSx.wsTrust.x200512.RequestSecurityTokenResponseCollectionDocument For a connection with WS-Trust v1.2, #ResponseDocument will be of type org.xmlsoap.schemas.ws.x2005.x02.trust.RequestSecurityTokenResponseDocument #Attributes org.sourceid.util.log.AttributeMap

SP Connections SAML 1.x message types and expressions

Message types	Available variables and classes/interfaces in Javadoc
AssertionType	#AssertionType org.sourceid.protocol.saml11.xml.AssertionType #AssertionTypes org.sourceid.protocol.saml11.xml.AssertionType[] #Attributes org.sourceid.util.log.AttributeMap
ResponseDocument	#ResponseDocument For a connection with WS-Trust v1.3, #ResponseDocument will be of type org.oasisOpen.docs.wsSx.wsTrust.x200512.RequestSecurityTokenResponseCollectionDocument For a connection with WS-Trust v1.2, #ResponseDocument will be of type org.xmlsoap.schemas.ws.x2005.x02.trust.RequestSecurityTokenResponseDocument #Attributes org.sourceid.util.log.AttributeMap

Example:

- Example of an AssertionType expression for SAML1.1.

```
#AssertionType.getAuthenticationStatementArray(0)
.getSubject().getNameIdentifier().setStringValue("JoeSAML2IDP"),
#AssertionType
```

- Example of a ResponseDocument expression for a connection with WS-Trust v1.3.

```
#RequestSecurityTokenResponseCollectionDocument.getRequestSecurityTokenResponseCollection()
.getRequestSecurityTokenResponseArray(0).setContext(‘context1’){code}
```

- Example of a ResponseDocument expression for a connection with WS-Trust v1.2.

```
#RequestSecurityTokenResponseDocument.getRequestSecurityTokenResponse().setContext('context1')
```

7. Click Next.

Setting a token lifetime

Specify a token's timeframe of validity before and after issuance.

About this task

Standards require a window of time during which a security token is considered valid. Each token has a time-stamp XML element as well as elements indicating the allowable lifetime of the token, in minutes, before and after the token time stamp.

Steps

1. Go to Applications > Integration > SP Connections.

2. In the WS-Trust STS window, click the Token Lifetime tab.

3. Override the default values for the fields in the following table.

Token timeframe parameters and descriptions

Field	Description
Minutes Before	The amount of time before the SAML token was issued during which it is to be considered valid. The default value is 5.
Minutes After	The amount of time after the SAML token was issued during which it is to be considered valid. The default value is 30.

4. Click Next.

Configuring token creation

The PingFederate security token service (STS) requires creating tokens to enable web services access to resources at your service provider (SP) partner's site.

About this task

For the PingFederate STS to issue a security token in response to requests for partner services, you must indicate what user attributes are to be included in the token attribute contract. The attribute values sent in the token are then derived by mapping those available from the token processor you select. As with browser single sign-on (SSO), the mapping can be augmented using local data stores, variable or constant text, or expressions.

Steps

- On the Token Creation, click Configure Token Creation to begin a token creation configuration.

Defining an attribute contract for IdP STS

During token creation configuration, define an attribute contract that the server sends in the security tokens issued in response to a web service client at your site.

About this task

An attribute contract is the set of user attributes that a web service client at your site expects to receive in security tokens issued for this connection. You identify these attributes on the Attribute Contract tab. For more information, see Attribute contracts.

Steps

1. Enter the attribute name in the Extend the Contract field. Attribute names are case-sensitive and must correspond to the attribute names, including claims, expected by the requesting web services client (WSC).

   Result:

   > **Tip**
   >
   > The Format attribute associated with the `NameID` element in outgoing SAML tokens can be set by adding an attribute called `SAML_NAME_FORMAT` . The value of that attribute can then be mapped later. For more information, see Configuring contract fulfillment for token creation.
   > For information about the `NameID` elements and applicable URI values, locate the SAML 2.0 specification at www.oasis-open.org/standards.

   > **Tip**
   >
   > You can add a special attribute, `SAML_AUTHN_CTX` , to indicate to the service provider (SP) the type of credentials used to authenticate to the identity provider (IdP) application-authentication context. Map a value for the authentication context on the attribute-mapping window later in the configuration, from any available attribute source, including the RST if a requested context is specified as a request parameter. For more information, see Configuring contract fulfillment for token creation.

2. Optional: For SAML 1.1 tokens, select a attribute namespace from the list.

   This field appears only when the chosen default token type is SAML 1.1 or SAML 1.1 for Office 365 in the WS-Trust STS > Protocol Settings configuration.

   Change the default namespace selection if you and your SP partner have agreed to a specific namespace.

   > **Note**
   >
   > You can customize name-format alternatives in the `custom-name-formats.xml` configuration file located in the `<pf_install>/pingfederate/server/default/data/config-store` directory. You must restart PingFederate to activate any changes made to this file.

   For more information about attribute namespace, see Attribute contracts.

3. Click Add.

4. Repeat until all applicable attributes are defined.

5. Click Next.

Result

Use the Edit, Update, and Cancel workflow to make or undo a change to an item. Use the Delete and Undelete workflow to remove an item or cancel the removal request.

Selecting a request contract

Select a request contract to be used to map attribute values into the security token.

About this task

This optional setting allows you to use XML parameters contained in RSTs for token-attribute mapping. For more information, see Managing STS request parameters. If you are not using request parameters, click Next to continue.

> **Note**
>
> If you are editing an existing configuration, you can change the request contract or disable this optional setting. These changes night require additional configuration changes in subsequent tasks.

Steps

1. To use request parameters, select the Request Contract check box to access the Request Contractlist.
2. Select a request contract from the Request Contract list.

   If the contract you want is in the Request Contract list, click Manage STS Request Parameters.

   When selected, you can choose the request contract as the attribute source in the IdP Token Processor Mapping configuration later in the setup.
3. Click Next.

Managing IdP token processor mappings

Identity provider (IdP) token processors are responsible for validating incoming security tokens as part of an STS operation. A configured and deployed token processor in PingFederate is known as a token processor instance. Map, edit, remove, or save a token processor instance.

About this task

You can map one or more token processor instances into an service provider (SP) connection to satisfy multiple session-management requirements where needed. The same token processor instances can be mapped in multiple SP connections.

When token processor instances are restricted to certain virtual server IDs, the allowed IDs are displayed in the Virtual Server IDs column.

Steps

1. In the Token Creation window, click the IdP Token Processor Mapping tab.
2. To map a token processor instance, click Map New Token Processor Instance.

   Choose from:

   - To edit the mapping configuration of a token process instance, open it by clicking on its name, select the setting that you want to reconfigure, and complete the change.
   - To remove a token processor instance or cancel the removal request, click Delete followed by Save or Undelete.
3. If you are creating a new connection and you are finished with mapping configuration, click Done.
4. If you are editing an existing configuration and want to keep your changes, click Save.

Selecting a token processor instance

Select an IdP token processor instance that can be used to authenticate users for a partner. Attributes returned by the token processor instance you choose for the token processor contract can be used to fulfill the attribute contract with your partner.

About this task

On the Token Processor Instance tab, choose an instance of a deployed token processor that suits your requirements for this connection.

> **Note**
>
> If you are editing a currently mapped token processor instance, you can toggle the **Override Instance Settings** check box. Clearing it removes all previously overridden settings for this connection. Selecting it provides you the opportunity to customize token processor settings specifically for this connection.

Steps

1. From the Token Processor Instance list, select a token processor instance.

   > **Note**
   >
   > If you do not see the desired token processor instance, click **Manage Token Processor Instances** to create a new instance of any deployed token processor.

2. Select the Override Instance Settings check box if you want to customize one or more token processor settings for this connection alone.

   > **Tip**
   >
   > Alternatively, you can create child token processor instances of a base token processor instance with overrides so that such customized settings can be applied to several connections. For more information, see Hierarchical plugin configurations.

   Result:

   When selected, the administrative console adds a new Override Instance tab.

Overriding a token processor instance

Customize the token processor instance settings for overriding one or more token processor settings for a connection.

About this task

On the Override Instance tab, configure override token processor settings specifically for this connection.

> **Note**
>
> Any changes to the base token processor instance are propagated to a connection provided the same changes are not overridden for the connection.

Steps

1. Click Override Instance Settings.
2. On the Instance Configuration, Extended Contract, and Token Attributes tabs, for each of the settings, select the Override check box and make your changes.

   > **Note**
   >
   > If you are editing a currently mapped token processor instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You can also remove all overridden settings on a per-tab basis by clearing the **Override** check box near the top of the window.
3. Click Next.
4. Click Done to close the Override Instance Settings window and continue with the rest of the IdP Token Processor Mapping window configuration.

Restricting a token processor to certain virtual server IDs

Virtual server IDs provide more configuration flexibility in cases where you need to identify your server differently when connecting to a partner in one connection for multiple environments or in multiple connections where the partner also supports multiple federation IDs. When you multiplex one connection for multiple environments, you can restrict each token processor added to a WS-Trust STS SP connection or IdP connection.

About this task

When you multiplex one connection for multiple environments see Multiple virtual server IDs, you can enforce authentication requirements by restricting a token processor to certain virtual server IDs on the Virtual Server IDs tab. By default, no restriction is imposed.

> **Note**
>
> If you are editing a currently mapped token processor instance, you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Steps

1. In the IdP Token Processor Mapping configuration window, go to the Virtual Server IDs tab.

   Select the Restrict Virtual Server IDs check box.

   Select one or more virtual server IDs that you want to allow for this token processor.

Selecting an attribute retrieval method for token creation

For token creation, you can query local data stores to help fulfill the attribute contract in conjunction with attribute values supplied by the token processor you are using with PingFederate.

About this task

The values supplied by the token processor are shown on the Attribute Retrieval tab, in the Token Processor Contract section.

To determine whether you need to look up additional values, compare the attribute contract against the token processor contract or the request contract, if configured. If the attribute contract requires more information, determine whether local data stores can supply it.

> **Note**
>
> If you are editing a currently mapped token processor instance, you can change the mapping method, which may require additional configuration changes in subsequent tasks.

Steps

1. Go to the IdP Token Processor Mapping window.
2. On the Attribute Retrieval tab, in the Token Processor Contract section, choose from the following options.

   Choose from:

   - Select the Retrieve Additional Attributes from Data Stores to Fulfill the Attribute Contract option if you want to configure one or more data stores to look up attribute for a single mapping.
   - Select the Use only the Token Processor Contract Values in the Outgoing Token option if you do not require data store query.
3. When finished, click Next.

Configuring attribute sources and user lookup for token creation

Specify a series of local data stores from which data, along with the attributes supplied in the incoming token, will be used to fulfill the attribute contract.

About this task

Attribute sources are specific data store or directory locations containing information that might be needed for the attribute contract. They are used to retrieve supplemental attributes. You can use more than one attribute source when mapping values to the attribute contract. The order matters and affects the queries differently. For example, if you plan on using the result of a query as an input to a subsequent query, stack your attribute sources accordingly.

> **Note**
>
> If you are editing a currently mapped token processor instance, you can add, remove, or reorder attribute sources, which might require additional configuration changes in subsequent tasks.

Steps

1. In the IdP Token Processor Mapping window, click the Attribute Sources & User Lookup tab.

> **Note**
>
> The **Attribute Sources & User Lookup** tab is only visible if you selected the **Retrieve Additional Attributes from Data Stores to Fulfill the Attribute Contract** option on the **Attribute Retrieval** tab. For more information, see Selecting an attribute retrieval method for token creation.

2. Click Add Attribute Source.

   Result:

   The Attribute Sources & User Lookup window configuration opens.

3. On the Data Store tab, choose a data store for PingFederate to look up attributes.
4. Enter a description in the Attribute Source Description field and a source ID in the Attribute Source ID field, if prompted, for the data store.
5. From the Active Data Store list, select a data store instance.

> **Tip**
>
> If the data store you want is not shown in the **Active Data Store** list, click **Manage Data Stores** to review or add a data store instance.

6. Depending on the data store type, the rest of the setup varies as follows.

Data store types and their required tasks

Data store type	Required tasks
JDBC	◦ Specifying database tables and columns ◦ Entering a database search filter
LDAP	◦ Specifying directory properties and attributes ◦ Defining encoding for binary attributes (optional) ◦ Entering a directory search filter
Other	◦ Specifying data source filters and fields

7. Repeat steps 2 - 6 as needed.
8. Click Save to exit the Attribute Sources & User Lookup window configuration.

Configuring contract fulfillment for token creation

Map values to the attributes defined for the contract. These are the values that are included in the SAML security tokens sent to the service provider (SP).

Steps

1. In the Token Creation | IdP Token Processor Mapping window, on the Attribute Contract Fulfillment tab, for each attribute, select a source from the Source list and then choose or enter a value. You must map all target attributes.
   - Token

When selected, the Value list is populated with attributes from the token processor instance. Select the desired attribute from the list. At runtime, the attribute value from the token processor instance is mapped to the value of the attribute in the SAML security token.

For example, to map the value of the Username Token Processor's `username` attribute as the value of the `TOKEN_SUBJECT` attribute on the contract, select Token from the Source list and username from the Value list.

- Context

When selected, the Value list is populated with the available context of the transaction. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the SAML security token.

> **Note**
>
> The **HTTP Request** and **STS SSL Client Certificate Chain** context values are retrieved as Java objects rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values, such as **Expression**.

> **Note**
>
> When using the **STS Basic Authentication Username**, **STS SSL Client Certificate's Subject DN**, or **STS SSL Client Certificate Chain** contexts, ensure the associated authentication is enabled and configured on the **System** > **Server** > **Protocol Settings** > **WS-Trust STS Settings** tab.

- Request

When selected, the Value list populates with parameter values from the token request received from the web service client. This selection is available only if a request contract was selected earlier on the Request Contract tab. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the SAML security token.

- LDAP, JDBC, or Other

When selected, the Value list populates with attributes that you have selected in the Attribute Source & User Lookup window configuration. Select the desired attribute from the list. At runtime, the attribute value from the attribute source is mapped to the value of the attribute in the SAML security token.

  - Expression

When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select Expression from the Source list, click Edit under Actions, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see Text.

+ Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- No Mapping

Select this option to ignore the Value field.

- Text

When selected, the text you enter is mapped to the value of the attribute in the single sign-on tokens at runtime. You can mix text with references to any of the values from the authentication source using the `$\{attribute}` syntax.

You can also enter values from your data store, when applicable, using the following syntax.

```
[.codeph]``$\{ds.[.varname]__attr-source-id.attribute__}``
```

where `attr-source-id` is the attribute source ID value and `attribute` is any of the selected attributes in the attribute source configuration.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

> **Note**
>
> If you are editing a currently mapped token processor instance, you can update the mapping configuration, which might require additional configuration changes in subsequent tasks.

Defining issuance criteria for token creation
About this task

On the Issuance Criteria tab, define the criteria that must be satisfied in order for PingFederate to process a request further. This token authorization feature provides the capability to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by adding a criterion. Choose the source that contains the attribute to be verified. Some sources, such as Mapped Attributes, are common to almost all use cases. Other sources, such as JDBC, depend on the type of configuration. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to be verified. Depending on the selected source, the available attributes or properties vary. Finally, specify the comparison method and the desired, compared-to, value.

If you define multiple criteria, all criteria must be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value depending on the chosen comparison method. The multi-value contains and multi-value does not contain comparison methods are intended for attributes that might contain multiple values. Such criterion is considered satisfied if one of the multiple values matches or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Important**
>
> When you multiplex one connection for multiple environments, consider using attribute mapping expressions to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access. For more information, see Multiple virtual server IDs and Issuance criteria and multiple virtual server IDs.

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. In the Token Generator Mapping & User Lookup configuration window, go to the Issuance Criteria tab.

2. In the Source list, select the attribute's source.

Depending on the selection, the Attribute Name list populates with associated attributes. See the following table for more information.

+

Attributes or properties and descriptions

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** and **STS SSL Client Certificate Chain** context values are retrieved as a Java object rather than text, use attribute mapping expressions to evaluate and return values.
JDBC, LDAP, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Token	Select to evaluate attributes from the token processor instance.

1. Select the attribute to be evaluated under Attribute Name.

> **Note**
>
> To evaluate the **STS Basic Authentication Username**, **STS SSL Client Certificate Chain**, or **STS SSL Client Certificate's Subject DN** context value, ensure that the associated authentication is enabled and configured on **System** > **Server** > **Protocol Settings** to open the **WS-Trust STS Settings** window.

2. In the Condition list, select the comparison method.

   Available methods:

- equal to
- equal to (case insensitive)
- equal to DN
- not equal to
- not equal to (case insensitive)
- not equal to DN
- multi-value contains

- **multi-value contains (case insensitive)**
- **multi-value contains DN**
- **multi-value does not contain**
- **multi-value does not contain (case insensitive)**
- **multi-value does not contain DN**

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value ...** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+

> **Note**
>
> To evaluate the **STS SSL Client Certificate's Subject DN** context value, you must select one of the **... DN** conditions. These methods normalize the DN before comparison to accommodate for different string representations that are still considered equivalent, such as case sensitivity or white space.

+ . In the Value field, enter the comparison value.

+

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

1. **In the Error Result field, enter a custom error message.**

   The Error Result field is used by the `faultstring` element for SOAP 1.1 and the `Reason/Text` element for SOAP 1.2. For more information on SOAP, see .w3.org/TR/2000/NOTE-SOAP-20000508///[Simple Object Access Protocol].

   Using an error code in the Error Result field allows an application to process the code in a variety of ways, such as displaying an error message or e-mailing an administrator.

   To use localized descriptions, enter a unique alias in the Error Result field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

   If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

2. **Click Add.**
3. **Optional: Repeat to add more criteria.**
4. **If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.**

For more information, see Attribute mapping expressions.

1. Click Show Advanced Criteria.
2. In the Expression field, enter the required expressions.
3. In the Error Result field, enter an error code or message.

> **Note**
>
> If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click Add.
5. Optional: Click Test, enter values in the applicable fields, and verify the results.
6. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Related links

- Token authorization
- Localizing messages for end users
- Attribute mapping expressions

Reviewing the IdP token processor mapping

Review the identity provider (IdP) token processor mapping configuration to make and save changes as needed.

About this task

On the Summary tab, review, amend, save, or discard your configuration changes.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click Done and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click Cancel.

Selecting a request error handling method

If you are using request parameters to fulfill the attribute contract and the parameter values are not supplied in the request token (RST) messages, you can choose whether to continue or abort the token-creation process.

About this task

On the Error Handling tab, select a request error handling method.

> **Note**
>
> On the **Request Contract** tab, you must select the **Request Contract** check box and configure a request contract for this connection for the **Error Handling** tab to appear after the **IdP Token Processor Mapping** tab.

Steps

1. Go to Applications > Integrations > SP Connections to choose the appropriate connection.
2. On the WS-Trust STS tab, click Configure WS-Trust STS.
3. On the Token Creation tab, click Configure Token Creation.
4. On the Error Handling tab, select one of the following options.

   Choose from:

   - To allow the STS transaction to continue with null attribute values sent in the generated token, select Send User to SP Using Null Values for Attributes.
   - To abort the STS transaction with null attribute values sent in the generated token, select Abort the STS Transaction.
5. Click Next.

Reviewing the token creation configuration

Review the token creation configuration to make changes as needed.

About this task

On the Summary tab, review, amend, save, or discard your token creation configuration changes.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click Done and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click Cancel.

Reviewing the IdP STS settings

Review the identity provider (IdP) security token service (STS) settings to make and save changes as needed.

About this task

On the Summary tab, review, amend, save, or discard the IdP STS settings.

Steps

1.
   - To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
   - To keep your changes, click Done and continue with the rest of the configuration.

     > **Tip**
     >
     > When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

   - To discard your changes, click Cancel.

Service provider STS configuration

This section covers the service provider (SP) configuration for security token service (STS).

Managing token generators

The PingFederate security token service (STS) uses token generators to issue security tokens that can be consumed by web services at your site. You must configure at least one generator to set up an STS connection or a token-to-token mapping.

About this task

In the Token Generators window, create, modify, review, or remove a token generator instance.

> **Note**
>
> PingFederate comes bundled with the SAML 1.1 Token Generator and SAML 2.0 Token Generator.
> You can deploy additional token translators from Ping Identity website.
> For simplicity, this topic focuses on configuring an instance of the SAML 1.1 or 2.0 Token Generator. For information about add-on token generators, see Integration overview.
> For more information about WS-Trust, see Web services standards.

Steps

1. Go to Applications > Token Exchange > Token Generators.

2. In the Token Generators window, choose from the following options.

Option	Description
Configure a new instance	Click Create New Instance
Modify an existing instance	Click the name of instance in the Instance Name column
View the usage of an existing instance	Click Check Usage in the Action column on the instance's row
Remove an existing instance	Click Delete in the Action column on the instance's row

> ⓘ **Note**
>
> By default, PingFederate automatically checks multi-connection errors whenever you access this window. This verifies that configured connections are not adversely affected by changes made here.
> If you experience noticeable delays in accessing this window, you can disable automatic connection validation. Go to **System** > **Server** > **General Settings**.

Selecting a token generator type

Token generators issue tokens for security token service and OAuth token exchange use cases. Select instances of a token generator type to use in mapping and policy configuration.

About this task

The first step in creating a token-generator instance is choosing the generator type.

Steps

1. Go to Applications > Token Exchange > Token Generators.

1. Select an existing generator instance by clicking its name or click Create New Instance to open the Create Token Generator Instance window.

2. On the Type tab, enter a name in the Instance Name field and ID in the Instance ID field for the token generator instance.

3. From the Typelist, select the token-generator type.

4. From the Parent Instance list, select an option.

   This is useful when you are creating an instance that is similar to an existing instance. The child instance inherits the configuration of its parent. You have the option to override one or more settings during the rest of the setup. Select the Override ... check box and make the adjustments as needed in one or more subsequent tabs.

5. Click Next.

Configuring a token generator instance

Configure the SAML token generator instance parameters depending on the use case requirements.

About this task

Depending on the selected token generator, the Instance Configuration tab presents you with different parameters.

Steps

1. Go to Applications > Token Exchange > Token Generators.
2. Select an existing generator instance by clicking its name or click Create New Instance to open the Create Token Generator Instance window.
3. On theInstance Configuration tab, configure the parameters for this instance type. For the integrated SAML 1.0 and 2.0 Token Generators, see the following table and specify parameters for generated SAML tokens.

SAML token generator instance field names and descriptions

Field	Instructions
Minutes Before	Enter a numerical value. This element in a SAML token allows for any server clock variability.
Minutes After	Enter a numerical value. This element in a SAML token allows for any server clock variability.
Issuer	Enter your SAML 2.0 entity ID or the SAML 1.x issuer as configured in the System > Server > Protocol Settings window.
Signing Certificate	Responses containing SAML tokens must be signed. Select a signing certificate from the list. If you have not yet created or imported your certificate into PingFederate, click Manage Signing Certificates. For more information, see Manage digital signing certificates and decryption keys.
Signing Algorithm	Select the signing algorithm corresponding to the selected certificate. Choices include SHA1 for both RSA and DSA. For a list of the available signing algorithms and their URIs, see Signing algorithms.
Include Certificate in KeyInfo	If selected, the entire public certificate is included with the assertion. Otherwise, a short hash reference to the certificate is sent.
Include Raw Key in KeyValue	If selected, the raw key is included in the `KeyInfo` element as well.
Audience	A unique identifier for the target web service, used for the `audience` element of the generated SAML token.

Field	Instructions
Confirmation Method	Choose from among available methods: ◦ urn...cm:sender-vouches Default option. ◦ urn...cm:bearer ◦ urn...cm:holder-of-key For more information, see WSS SAML Token Profile.
Encryption Certificate	The web service provider's public certificate for encryption is required only if holder-of-key is selected as the confirmation method. Select a partner certificate from the list. If you have not yet imported the certificate from your partner, click Manage Certificates to do so. For more information, see Managing certificates from partners.

Message Customization expression	Click Show Advanced Fields to see this field.
	An OGNL expression to customize the assertion. The returned type from the expression must be an AssertionType, or the customization will be ignored.
	The available attributes are:
	◦ `#AssertionType`: `org.sourceid.saml20.xmlbinding.assertion.AssertionType`
	◦ `#Attributes`: `org.sourceid.util.log.AttributeMap`
	The following example is for SAML2. The line breaks are provided to improve readability.
	`#AssertionType`
	`  .getSubject()`
	`  .getNameID()`
	`  .setStringValue("JoeSAML2IDP"),`
	`#AssertionType`
	The following example is for SAML1.1.
	`#AssertionType`
	`  .getAuthenticationStatementArray(0)`
	`  .getSubject().getNameIdentifier()`
	`  .setStringValue("Joe123"),`
	`#AssertionType`

For information about add-on generators, see Integration overview.

4. Click Next.

Extending a token generator contract

Token generators allow administrators to add to a built-in list of user attributes that the generator includes in the outgoing token—an extended generator-attribute contract.

About this task

The Extended Contract tab shows a different list of attributes in the Core Contract section, depending on the token generator selected.

Steps

1. Go to Applications > Token Exchange > Token Generators.
2. Select an existing generator instance by clicking its name or click Create New Instance to open the Create Token Generator Instance window.
3. On the Extended Contract tab, in the Extend the Contract field, enter a name of the desired attribute. Click Add.

   Repeat this step as needed to add another attribute.
4. Click Next.

Reviewing the token generator configuration

Review, save, or make changes as needed to your token generator instance configuration.

About this task

On the Summary tab, review, amend, save, or discard your token generator instance configuration.

Steps

- To keep your changes, click Save.
- To amend your configuration, click the name of the corresponding tab and then follow the configuration wizard to complete the task.
- To discard your changes, click Cancel.

Configuring IdP connections for STS

Configure a security token services (STS) connection to an identity provider (IdP) partner either in conjunction with browser-based single sign-on (SSO) or independently.

Configuring protocol settings for SP STS

Configure the processing options for validating incoming SAML tokens in your identity provider (IdP) partner connection either in conjunction with browser-based single sign-on (SSO) or independently.

About this task

Select whether the STS should validate incoming tokens only or validate and then generate other types of tokens.

Steps

1. Go to Authentication > Integration > IdP Connections.
2. On the WS-Trust STS tab, click Configure WS-Trust STS.

3. On the Protocol Settings tab, from the Request Processing Options list, choose one of the following options:

   Choose from:

   - To only validate incoming SAML tokens, select Validate Incoming SAML Token.
   - To validate and then also generate local tokens to enable single sign-on (SSO) access to web services at your site, select Validate Incoming SAML Token and Issue Local Token.

     > **Note**
     >
     > If you choose to generate local tokens as well, you must set up at least one token generator.

Configuring token generation

About this task

Details of this configuration are handled under the Token Generation configuration.

Steps

- Go to Authentication > Integration > IdP Connections.
- On the WS-Trust STS tab, click Configure WS-Trust STS.
- On the Token Generation tab, click Configure Token Generation.

  Result:

  The Token Generation window opens.

Defining an attribute contract for SP STS

An attribute contract is the set of user attributes expected in incoming SAML assertions. For more information, see Attribute contracts.

About this task

On the Attribute Contract tab, identify the user attributes.

Optionally, you can mask the values of attributes, other than `SAML_SUBJECT`, in logs that PingFederate writes when it receives security tokens.

> **Note**
>
> Use the **Edit**, **Update**, and **Cancel** workflows to make or undo a change to an item. Use the **Delete** and **Undelete** workflows to remove an item or cancel the removal request.

Steps

1. Go to Authentication > Integration > IdP Connections.

2. On the WS-Trust STS tab, click Configure WS-Trust STS.
3. On the Token Generation tab, click Configure Token Generation.

   Result:

   The Token Generation configuration window opens.
4. Click the Attribute Contract tab.
5. Enter the name in the Extend the Contract field.

   > ⓘ **Note**
   >
   > Attribute names are case-sensitive and must correspond to the attribute names expected by the requester.
6. Optional: Select the Mask Values in Log check box .
7. Click Add.
8. Repeat until all applicable attributes are defined.
9. Click Next.

Managing SP token generator mappings

Token generators provide a mechanism through which PingFederate can generate a local token based upon an incoming SAML token, including mapping user attributes to be included in the generated token. A configured and deployed token generator in PingFederate is known as a token generator instance.

About this task

In the Token Generator Mapping & User Lookup configuration window, manage service provider (SP) token generator mappings.

Map one or more token generator instances into an identity provider (IdP) connection to satisfy different token requirements by the web services at your site. The same token generator instances can be mapped in multiple connections.

When token generator instances are restricted to certain virtual server IDs, the allowed IDs are displayed under Virtual Server IDs.

Steps

1. Go to Authentication > Integration > IdP Connections > WS-Trust STS.
2. Click Configure WS-Trust STS.

   Result:

   The WS-Trust STS configuration window opens.
3. On the Protocol Settings tab, from the Request Processing Options list, select Validate Incoming SAML Token and Local Issue Token. Click Next.

Result:

This will add a Token Generation tab.

4. On the Token Generation tab, click Configure Token Generation.

   Result:

   The Token Generation configuration window opens.

5. On the Token Generator Mapping & User Lookuptab, click Map New Token Generator Instance to open the Token Generator Mapping & User Lookup configuration window.

   Choose from:

   - To map a token generator instance, click Map New Token Generator Instance.
   - To edit the mapping configuration of a token generator instance, on the Token Generator Instance tab, click Manage Token Generator to open the Token Generators window. Click the token generator instance in the Instance Name section to open the configuration summary for this token generator instance. Select the setting that you want to reconfigure, and complete the change by clicking Done.
   - To remove a token generator instance or cancel the removal request, click Delete followed by Save or Undelete.

6. If you are creating a new connection and you are finished with mapping configuration, click Done. If you are editing an existing configuration and want to keep your changes, click Save.

Selecting a token generator instance

Select the token generator instance you would like to activate for incoming SAML messages from the service provider (SP) partner.

About this task

> ⓘ **Note**
>
> If you are editing a currently mapped token generator instance, you can toggle the **Override Instance Settings** setting. Clearing it removes all previously overridden settings for this connection. Selecting it provides you the opportunity to customize token processor settings specifically for this connection.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance.

   Result:

   The Token Generator Mapping & User Lookup configuration window opens.

2. On the Token Generator Instance tab, select an instance from the Token Generator Instance.

   > ⓘ **Note**
   >
   > If you do not see the desired token generator instance, click **Manage Token Generator Instances** to create a new instance of any deployed token generator.

3. Select the Override Instance Settings check box to customize one or more token processor settings for this connection alone.

   Result:

   When selected, the administrative console adds a Override Instance tab and a its new set of sub tasks.

   > **Tip**
   >
   > You can also create child token processor instances of a base token processor instance with overrides so that the customized settings can be applied to several connections. For more information, see Hierarchical plugin configurations.

4. Click Next.

Overriding a token generator instance

Make changes to the token generator instance to override it or to leave it as is and propagate it for a particular service provider connection.

About this task

Override token generator settings for a specific service provider connection.

> **Note**
>
> Any changes to the base token generator instance are propagated to a connection provided the same changes are not overridden for the connection.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance.

   Result:

   The Token Generator Mapping & User Lookup configuration window opens.

2. On the Override Instance tab, select the Override Instance Settings check box.

   > **Note**
   >
   > The override setting windows are functionally identical to those used for creating a new token generator instance. For more information, see Managing token generators.

3. On each of the settings windows, select the Override check box, make your changes, and then click Next.

4. When you are finished, click Done to continue with the rest of the mapping configuration.

Restricting a token generator to certain virtual server IDs

Virtual server IDs provide more configuration flexibility in cases where you need to identify your server differently when connecting to a partner in one connection for multiple environments or in multiple connections where the partner also supports multiple federation IDs.

About this task

When you multiplex one connection for multiple environments, you can enforce integration requirements by restricting a token generator to certain virtual server IDs on the Virtual Server IDs tab. By default, no restriction is imposed. For more information, see Multiple virtual server IDs.

> **Note**
>
> If you are editing a currently mapped token generator instance, you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance to open the Token Generator Mapping & User Lookup configuration window.
2. Click the Virtual Server IDs tab.
3. Select the Restrict Virtual Server IDs check box.
4. Select one or more virtual server IDs that you want to allow for this token generator.

Selecting an attribute retrieval method for token generation

You can fulfill the token generator contract by using only the attributes from the incoming SAML token or by using these attributes to look up additional information from a local data store.

About this task

For token generation, you can query local data stores to help fulfill the token generator contract, in conjunction with attribute values supplied by the incoming token.

The values supplied by the token are shown in the Attribute Contract section on the Attribute Retrieval tab.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance.

   Result:

   The Token Generator Mapping & User Lookup configuration window opens.

2. On the Attribute Retrieval tab, select how you want to fulfill the token generator contract for an instance.

   Choose from:

   - If the incoming SAML token contains all the attributes that your application requires, select Use only the attributes available in the incoming token.
   - To set up a data store query, select Use the incoming token to look up additional information and then follow a series of sub tasks to complete the configuration.

   For step-by-step instructions, see Choosing a datastore.

   > **Note**
   >
   > If you are editing a currently mapped token generator instance, you can change the mapping method, which might require additional configuration changes in subsequent tasks.

Configuring contract fulfillment for token generation

Fulfill your token generator contract requirements with values from the incoming SAML token, dynamic text, expressions, or from a data store lookup.

About this task

Map the values that the web services require to the attributes defined for the contract.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance.

   Result:

   The Token Generator Mapping & User Lookup configuration window opens.

2. On the Token Generator Contract Fulfillment tab, for each attribute, select a source from the Source list and then choose or enter a value. You must map all attributes.

   - Assertion

     When selected, the Value list populates with attributes from the incoming SAML token (assertion). Select the desired attribute from the list. At runtime, the attribute value from the assertion is mapped to the value of the attribute in the local token.

     For example, to map the value of `TOKEN_SUBJECT` from a SAML assertion as the value of the `subject` user identifier on the token generator contract, select Assertion from the Source list and TOKEN_SUBJECT from the Value list.

   - Context

     When selected, the Value list populates with the available context of the transaction. Select the desired context from the list. At runtime, the context value is mapped to the value of the attribute in the local token.

     > **Note**
     >
     > Because the **HTTP Request** and **STS SSL Client Certificate Chain** context values are retrieved as Java objects rather than text, use OGNL expressions to evaluate and return values, sees **Expression**.

     > **Note**
     >
     > When using the **STS Basic Authentication Username**, **STS SSL Client Certificate's Subject DN**, or **STS SSL Client Certificate Chain** contexts, ensure the associated authentication is enabled and configured on the **System** > **Server** > **Protocol Settings** > **WS-Trust STS Settings** tab.

   - LDAP, JDBC, or Other

     When selected, the Value list populates with attributes that you have selected from the data store. Select the desired attribute from the list. At runtime, the attribute value from the data store is mapped to the value of the attribute in the local token.

     - Expression

When enabled, this option provides more complex mapping capabilities, such as transforming incoming values into different formats. Select Expression from the Source list, click Edit under Actions, and compose your OGNL expressions. All variables available for text entries are also available for expressions. For more information, see Text.

+ Expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see Attribute mapping expressions.

- No Mapping

  Select this option to ignore the Value field.

- Text

  When selected, the text you enter is used at runtime. You can mix text with references to any of the values from the SAML token, using the `$\{attribute}` syntax.

  You can also enter values from your datastore, when applicable, using this syntax:

  `[.codeph]``$\{ds.[.varname]__attribute__}```

  where `attribute` is any of the attributes that you have selected from the data store.

> **Tip**
>
> You can reference attribute values in the form of `$\{attributeName:-defaultValue}`. The default value is optional. When specified, it is used at runtime if the attribute value is not available. Do not use `$\{` and `}` in the default value.

> **Note**
>
> If you are editing a currently mapped token generator instance, you can update the mapping configuration, which might require additional configuration changes in subsequent tasks.

3. Click Next.

Defining issuance criteria for token generation

PingFederate can evaluate various criteria to determine whether users are authorized to access service provider (SP) resources.

About this task

On the Issuance Criteria tab, define the criteria that must be satisfied in order for PingFederate to process a request further. This token authorization feature provides the capability to conditionally approve or reject requests based on individual attributes.

Begin this optional configuration by adding a criterion. Choose the source that contains the attribute to be verified. Some sources, such as Mapped Attributes, are common to almost all use cases. Other sources, such as JDBC, depend on the type of configuration. Irrelevant sources are automatically hidden. After you select a source, choose the attribute to be verified. Depending on the selected source, the available attributes or properties vary. Finally, specify the comparison method and the desired, compared-to, value.

If you define multiple criteria, all criteria must be satisfied for PingFederate to move a request to the next phase. A criterion is satisfied when the runtime value of the selected attribute matches or does not match the specified value depending on the chosen comparison method. The multi-value contains and multi-value does not contain comparison methods are intended for attributes that might contain multiple values. Such criterion is considered satisfied if one of the multiple values matches or does not match the specified value. Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions

> **Important**
>
> When you multiplex one connection for multiple environments, consider using attribute mapping expressions to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access. For more information, see Multiple virtual server IDs and Issuance criteria and multiple virtual server IDs.

> **Note**
>
> All criteria defined must be satisfied or evaluated as true for a request to move forward. As soon as one criterion fails, PingFederate rejects the request and returns an error message.

Steps

1. On the Token Generator Mapping & User Lookup tab, click Map New Token Generator Instance.

   The Token Generator Mapping & User Lookup configuration window opens.

2. Click the Issuance Criteria tab.

   Depending on the selection, the Attribute Name list populates with associated attributes. See the following table for more information.

Source entries and descriptions

Source	Description
Context	Select to evaluate properties returned from the context of the transaction at runtime. **Note** Because the **HTTP Request** and **STS SSL Client Certificate Chain** context values are retrieved as a Java object rather than text, use attribute mapping expressions to evaluate and return values.
JDBC, LDAP, or other types of datastore (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Token	Select to evaluate attributes from the incoming SAML token.

3. Select the attribute to be evaluated in the Attribute Name column.

> **Note**
>
> To evaluate the **STS Basic Authentication Username**, **STS SSL Client Certificate Chain**, or **STS SSL Client Certificate's Subject DN** context value, ensure that the associated authentication is enabled and configured on **System > Server > Protocol Settings** to open the **WS-Trust STS Settings** window.

Available methods:

- equal to
- equal to (case insensitive)
- equal to DN
- not equal to
- not equal to (case insensitive)
- not equal to DN
- multi-value contains
- multi-value contains (case insensitive)
- multi-value contains DN
- multi-value does not contain
- multi-value does not contain (case insensitive)
- multi-value does not contain DN

> **Note**
>
> The first six conditions are intended for single-value attributes. Use one of the **multi-value …** conditions for PingFederate to validate whether one of the attribute values matches the specified value. When an attribute has multiple values, using a single-value condition causes the criteria to fail.

+

> **Note**
>
> To evaluate the **STS SSL Client Certificate's Subject DN** context value, you must select one of the **… DN** conditions. These methods normalize the DN before comparison to accommodate for different string representations that are still considered equivalent, such as case sensitivity or white space.

> **Note**
>
> To evaluate the **STS SSL Client Certificate's Subject DN** context value, you must select one of the **… DN** conditions. These methods normalize the DN before comparison to accommodate for different string representations that are still considered equivalent, such as case sensitivity or white space.

1. In the Value field, enter the comparison value.

> **Note**
>
> Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. For more information, see Attribute mapping expressions.

The Error Result field is used by the `faultstring` element for SOAP 1.1 and the `Reason/Text` element for SOAP 1.2. For more information on SOAP, see the World Wide Web Consortium's .w3.org/TR/2000/NOTE-SOAP-20000508/// [Simple Object Access Protocol].

Using an error code in the Error Result field allows an application to process the code in a variety of ways, such as display an error message or e-mail an administrator.

To use localized descriptions, enter a unique alias in the Error Result field, such as `someIssuanceCriterionFailed`. Insert the same alias with the desired localized text in the applicable language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

If not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

+ . Click Add.

+ . Optional: Repeat to add more criteria.

+ . If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

+ For more information, see Attribute mapping expressions.

1. Click Show Advanced Criteria.
2. In the Expression field, enter the required expressions.
3. In the Error Result field, enter an error code or message.

> **Note**
>
> If the expressions resolve to a string value instead of `true` or `false`, the returned value overrides the **Error Result** field value.

4. Click Add.
5. Optional: Click Test, enter values in the applicable fields, and verify the results.
6. Optional: Repeat to add multiple criteria using attribute mapping expressions.

   1. When finished, click Next.

Related links
Reviewing the SP token generator mapping

Review the service provider (SP) token generator mapping instance.

About this task

On the Summary tab, review, amend, save, or discard your changes.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click Done and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click Cancel.

Reviewing the token generation configuration

Review the token generation configuration to make changes or save as needed.

About this task

On the Summary tab, review, amend, save, or discard your configuration.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click Done and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click Cancel.

Reviewing the SP STS configuration

Review the service provider (SP) security token service (STS) configuration to make changes or save as needed.

About this task

On the Summary tab, review, amend, save, or discard your changes.

Steps

- To amend your configuration, click the corresponding tab title, then follow the configuration wizard to complete the task.
- To keep your changes, click Done and continue with the rest of the configuration.

> **Tip**
>
> When editing an existing configuration, you can also click **Save** as soon as the administrative console offers the opportunity to do so.

- **To discard your changes, click Cancel.**

PingFederate Performance Tuning Guide

This section shows you to how to fine-tune a few simple application and system level settings to enable PingFederate to achieve maximum performance of the hardware chosen for your deployment.

The default configuration since PingFederate 10.2 is acceptable for most small size deployments. Mission-critical and high-transaction volume deployments might require additional tuning.

This guide addresses several areas of tuning such as logging, concurrency, memory, and Java-specific tuning options. It is not designed as a one-size-fits-all set of instructions to optimize PingFederate, but more as a checklist of suggestions for areas of the product that can be tuned to improve performance, and any tradeoffs associated with those changes. For ultimate reassurance that any fine-tuned settings will meet your expectations, performance testing in a lab environment is recommended.

Logging

This section explains the logging practices of PingFederate and discusses minimizing the system's overall performance impact.

Logging tracks various aspects of the system's overall performance and requires a certain amount of system resources, which affects the system's overall performance. In particular, writing to the log files takes the greatest amount of resources. To minimize the performance impact, PingFederate uses the high-performance asynchronous logger from Log4j 2 for logging runtime and administrative events, including status and error messages used for troubleshooting. To preserve transactional integrity, audit information logs synchronously.

Although the bulk of logging is executed asynchronously, decreasing the amount of information written to log files always provides the best possible performance.

PingFederate only records messages tagged with log level `INFO`, `WARN`, `ERROR`, and `FATAL` to the server log and the provisioner log. Messages with `DEBUG`, or `TRACE` tags, are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purposes, you can enable console logging or verbose messages.

> **Important**
>
> When you no longer require console logging or verbose messages, turn them off. On Windows, never highlight the console output because it might slow or stop PingFederate from processing requests.

Related links

- Enabling console logging

Operating system tuning

This section contains tuning recommendations for your operating system.

The tuning recommendations provided here work best in preventing deployment issues in high capacity environments.

Linux tuning

Follow these recommendations for your Linux environment to prevent deployment issues, to increase the performance and capacity of the networking stack, particularly TCP and the file descriptor usage, and to enable PingFederate to handle a high volume of concurrent requests.

Network/TCP tuning

For `SystemV`, add or modify the following entries in the `/etc/sysctl.conf` file.

For `systemd`, you can create a sysctl preload/configuration file in `/etc/sysctl.d` (for example, `99-sysctl.conf`) in which to add and modify the following entries.

```
# TCP Tuning #
# Controls the use of TCP syncookies (default is 1)
# and increase the number of outstanding syn requests allowed.
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_max_syn_backlog=8192

# Increase number of incoming connections.
# somaxconn defines the number of request_sock structures allocated
# per each listen call.
# The queue is persistent through the life of the listen socket.
net.core.somaxconn=4096

# Increase number of incoming connections backlog queue.
# Sets the maximum number of packets, queued on the INPUT side,
# when the interface receives packets faster
# than kernel can process them.
net.core.netdev_max_backlog=65536

# increase system IP port limits
net.ipv4.ip_local_port_range=2048 65535

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling=1

# decrease TCP timeout
net.ipv4.tcp_fin_timeout=10

# Allow reuse of sockets in TIME_WAIT state for new connections
# (While this may increase performance, use with caution according
# to the kernel documentation.  This setting should only be enabled
# after the system administrator reviews security considerations.)
net.ipv4.tcp_tw_reuse=1

# Increase the read and write buffer space allocatable
# (minimum size, initial size, and maximum size in bytes)
net.ipv4.tcp_rmem = 4096 65536 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# The maximum number of packets which may be queued
# for each unresolved address by other network layers
net.ipv4.neigh.default.unres_qlen=100
net.ipv4.neigh.eth0.unres_qlen=100
net.ipv4.neigh.em1.unres_qlen=100

# Default Socket Receive and Write Buffer
net.core.rmem_default=8388608
net.core.wmem_default=8388608
```

Increase file descriptor limits

Add or modify the following lines in the `/etc/security/limits.conf` file where *pf_user* is the user account used to run the PingFederate java process or `*` for all user accounts.

```
pf_user  soft nofile 10400
 pf_user  hard nofile 10400
```

Windows tuning

Follow these recommendations for your Windows environment to prevent deployment issues, to increase the performance and the capacity of the networking stack, specifically the TCP socket, and to enable PingFederate to handle a high volume of concurrent requests.

About this task

Use Command Prompt and a Registry Editor to edit the `cmd.exe` file and the `regedit.exe` file respectively.

Steps

1. Edit the `jvm-memory.options` file with a time stamp.
    1. Start the Command Prompt application `cmd.exe`.
    2. Enter `netsh int ipv4 show dynamicportrange tcp` to view the ephemeral ports.
    3. Enter `netsh int ipv4 set dynamicport tcp start=1025 num=64510` to increase the range of the ephemeral ports using administrative privileges.
    4. Reboot the server.
    5. Enter `netsh int ipv4 show dynamicportrange tcp` to confirm the updated port range.
2. Reduce socket TIME_WAIT delay.
    1. Start the Registry Editor application `regedit.exe`.
    2. Go to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`.
    3. Create a new DWORD 32 bit value and set the name as `TcpTimedWaitDelay`.
    4. Set a decimal value of `30`.
    5. Reboot the server.

Concurrency

This section describes how to configure PingFederate to support more concurrent requests to optimize your deployment.

The more requests processed in parallel, the more requests processed over all. Given the appropriate amount of hardware, processing *N* requests concurrently is typically faster than processing *N* requests serially.

In PingFederate, there are two main pools of threads that control the level of concurrent user requests: Acceptor Threads and Server Threads. Acceptor threads receive the HTTPS requests and pass those requests on to available server threads to be processed.

Caveats

This topic serves as a guideline for optimizing the concurrency of your deployment. On a large system with multiple CPUs, or cores, a thread pool that is too small will under-utilize the available processor resources. A thread pool that is too large can cause the system to become flooded and unusable.

A good target for the CPU is between 60%-80% utilization when under nominal, standard user load. This way CPU resources are not under-utilized while still allowing room for occasional load spikes. The level of concurrency in PingFederate might need to be decreased, or even increased, depending on the system's configuration, the adapters in use, available memory, and other processes competing for resources. All deployments are different. This section serves as a guideline of where to start when tuning the server.

Configuring the HTTP connection pool

Optimize connections with external services by adjusting the number and duration of connections.

About this task

PingFederate uses an HTTP connection pool to enable HTTP communication with external systems, including PingOne communication, LDAP gateways, some PingFederate adapters, and many other functions.

Most environments won't need to tune HTTP connection pool values.

If you find that you need to adjust timeout values, it might indicate issues with your network or overall capacity. You should investigate those configurations before you adjust these settings.

> **Tip**
>
> If you do need to tune the HTTP connection pool, you should adjust and test `max-connections` and `max-connections-per-route` first.
> Adjust `max-connections` to match the maximum `jetty-threads` value and `max-connections-per-route` to match the expected maximum external calls. For example, if you expect to have up to 200 threads using the PingOne API, set `max-connections-per-route` to 200.
> Test and adjust these values as necessary.

Steps

1. Open the `<pf_install>/pingfederate/server/default/data/config-store/http-connection-pooling-manager.xml` file.

2. Adjust the values as detailed in the following table:

Setting	Default Value	Description
`max-connections`	`350`	The maximum number of connections in the pool.
`max-connections-per-route`	`100`	The maximum number of connections per route (port or URL).

Setting	Default Value	Description
`connection-timeout`	`60000`	The time in milliseconds available to establish a connection with the remote host.
`request-timeout`	`120000`	The time in milliseconds available to retrieve a connection from the connection pool.
`connection-idle-timeout`	`30000`	The amount of time a connection in the pool can be idle before it is cleaned up in milliseconds.
`keepalive-timeout`	`1000`	The amount of time in milliseconds that a connection can idle before it is returned to the pool for use.
`cleanup-delay-secs`	`10`	The delay in seconds before cleaning the HTTP client cache.

3. Save and close the file.

4. Restart PingFederate

5. For a clustered PingFederate environment, replicate the configuration to your other nodes. Learn more in Replicating configurations.

Tuning the acceptor queue size

For optimal performance, particularly in larger deployments, PingFederate uses a non-blocking I/O model to process requests. You can tune the acceptor queue size parameters for your environment.

About this task

If the request queue reaches its maximum size, additional requests will receive a connection refused error. If this occurs in your environment, you can increase the values of the acceptor queue size parameters `pf.admin.acceptQueueSize` and `pf.runtime.acceptQueueSize`. The `pf.admin.acceptQueueSize` parameter applies to the administrative console and the `pf.runtime.acceptQueueSize` parameter applies to the engine nodes.

Steps

1. Stop PingFederate.

2. Open the `<pf_install>/pingfederate/bin/run.properties` file in an editor.

> **Tip**
>
> Consider making a backup copy of this file.

3. Go to the following section and change the values of the `pf.admin.acceptQueueSize` and `pf.runtime.acceptQueueSize` parameters.

   ```
   # HTTP Connector Queue Size Settings
   # ---------------------------------
   # The following properties control the queue size of the HTTP connector.
   #
   # Please refer to the performance tuning guide for further tuning guidance.
   pf.admin.acceptQueueSize=512
   pf.runtime.acceptQueueSize=512
   ```

4. Save your changes and restart PingFederate.

5. For a clustered PingFederate environment, repeat the previous steps on each engine node.

Tuning the server thread pool

When tuning the server thread pool, set the minimum and maximum number of threads to optimize PingFederate for your needs.

About this task

Set the minimum and maximum numbers according to the expected user load:

- Set the minimum number of threads to between 75% and 100% of the number of requests you expect the system to handle most often.

- Set the maximum number of threads to between 25% and 50% higher than the minimum to handle load spikes.

Performance testing offers an alternative guideline. Testing shows that PingFederate performs well when the server thread pool has 25 to 50 server threads per available CPU core, assuming sufficient memory. For example, if your PingFederate system has one CPU with four cores, the total available cores is four. In that case, the minimum thread value would be 100 and the maximum thread value would be 200.

> **Note**
>
> This alternative guideline might not apply to larger systems. For example, if PingFederate is running on a system with 24 CPU cores, it doesn’t make sense to size the thread pool at a minimum of 600 threads and a maximum of 1200 unless you expect to normally handle at least 800 concurrent requests.

For more information on managing memory for PingFederate, see Memory.

Steps

1. Stop PingFederate.

2. Edit the `<pf_install>/pingfederate/bin/run.properties` file.

   > **Tip**
   >
   > Consider making a backup copy of this file.

3. Go to the following section, and change the thread number.

```
# HTTP Server Thread Pool Settings
# --------------------------------
# The following properties control the minimum and the maximum number of threads used to configure
PingFederate thread pools.
#
# Please refer to the performance tuning guide for further tuning guidance.
pf.admin.threads.min=1
pf.admin.threads.max=10
pf.runtime.threads.min=10
pf.runtime.threads.max=200
```

4. Save your changes and restart PingFederate.

5. For a clustered PingFederate environment, repeat the previous steps on each engine node as needed.

Configuring connection pools to datastores

Java Database Connectivity (JDBC) and LDAP datastores use connection pooling to improve the performance and efficiency of communicating with external systems. For optimal performance, a number of connections are required to handle most or all the requests in parallel.

About this task

In the Data & Credential Stores page, set the minimum and maximum values for connection pools to JDBC and LDAP data stores.

Connection pools improve efficiency by maintaining persistent connections to the JDBC or LDAP server preventing the expense of creating the connection on demand. Connection pools also allow more control over the load placed on the back-end server. It might not be necessary to have a connection available for every concurrent request received by the server, but having too few available will cause requests to wait when accessing JDBC and LDAP resources.

> **Important**
>
> Size the connection pool based on the capacity and limitation of the database or LDAP server. Sizing the connection pool beyond the capability of the back-end server could lead to PingFederate flooding the datastore without any performance improvement. For optimal performance, size connection pools large enough to handle between 50% and 100% of the number of concurrent requests the server is expected to encounter often. Learn more about optimizing the connection pool in Best practices for tuning the JDBC connection pool in the Ping Identity Knowledge Base.

Steps

1. Choose from configuring connection pools to JDBC or LDAP datastores:

Datastore type	Configuration steps
Configuring connection pools to JDBC datastores	1. Go to System > Data & Credential Stores > Data Stores, and select the applicable JDBC datastore. 2. Go to Database Config > Advanced. 3. On the Advanced Database Options page: 1. Set the Minimum Pool Size value to 50% of the `maxThreads` value. 2. Set the Maximum Pool Size value to between 75% and 100% of the `maxThreads` value, subject to the capability of the back-end database server. **Note** The `maxThreads` value is defined in the `<pf_install>/pingfederate/etc/run.properties` file. Learn more in Tuning the server thread pool).
Configuring connection pools to LDAP datastores	1. Go to System > Data & Credential Stores > Data Stores, and select the applicable LDAP datastore. 2. Go to the LDAP Configuration > Advanced > Advanced LDAP Options. 3. Set the Minimum Connections value to 50% of the `maxThreads` value. 4. Set the Maximum Connections value to between 75% and 100% of the `maxThreads` value, subject to the capability of the back-end database server.

2. For a clustered PingFederate environment, replicate the changes to all engine nodes on the System > Server > Cluster Management page.

Snapshot isolation for high-volume transactions

Because deadlocks can occur in some tables in high-concurrency environments with contention for shared data, you can enable READ_COMMITTED_SNAPSHOT in SQL Server to improve the performance and scalability of applications with long-running transactions.

When snapshot isolation is enabled, each transaction reads data from a snapshot of the database taken at the start of the transaction rather than directly from the database itself.

Because there can be some overhead with snapshot isolation, evaluate and test this setting carefully before deploying in production environments. For more information on snapshot isolation and how to configure the option, see Snapshot Isolation in SQL Server in the Microsoft documentation.

Memory

After the CPU, memory is the most important resource for sizing Java virtual machine (JVM) heap, managing garbage collection, and optimizing the overall performance of your PingFederate deployment.

The Concurrency section describes how to configure PingFederate to support more concurrent requests. This section highlights how supporting increasing concurrency requests can affect PingFederate's performance because these requests require an increase in memory. Because PingFederate is a Java application, it is important to consider how tuning affects, or is affected by, garbage collection. This section is not a guide to garbage collection theory or ergonomics.

JVM heap

The most important tuning for the Java Virtual Machine (JVM) is the size of the heap memory, which ensures adequate memory is available to manage garbage collection and optimize overall performance.

If the demands require more memory than what is currently available, the JVM must grow the heap, if it can, or perform garbage collection to provide memory to allocate. Resizing the heap and garbage collecting can be an expensive processes and negatively impact performance. Sizing the heap to ensure an adequate amount of memory is available but still manageable to garbage collection is important to optimize overall performance.

PingFederate attempts to optimize JVM heap and garbage collector settings at the time of installation and upgrade. Regardless of available memory, PingFederate uses the Garbage-First (G1) garbage collector (GC). These settings assume PingFederate will be the only service on the server and consume a majority of the memory. Depending on your environment, you can override these settings at a later time.

Additional considerations

The JVM can grow the heap from the minimum heap variable value up to the maximum heap variable value. However, growing the heap is often an expensive exercise and requests memory from the operating system. In addition, the JVM must also reorganize the heap to account for the memory being added. To conserve memory in your deployment, set a lower value for the minimum heap than that of the maximum heap to ensure you are not reserving unused memory. If you have enough memory that a certain amount is easily earmarked for the PingFederate server, adjust the size of the heap by setting the minimum heap and maximum heap to the same value. This allows the JVM to reserve its entire heap and decrease the amount of resizing that the JVM needs to perform if the amount of memory in use exceeds the value of the minimum heap.

The memoryoptions utility

Where to find the `memoryoptions` utility in a PingFederate installation and how the utility's expected behavior differs in a Linux or Windows system.

PingFederate installation and upgrade tools use the **`memoryoptions`** utility to record the default options for the Java heap and the garbage collector (GC) in a configuration file. Regardless of available memory, PingFederate uses the Garbage-First (G1) GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. As needed, administrators can re-run the utility or manually edit the configuration file.

The **`memoryoptions`** utility, located in the `<pf_install>/pingfederate/bin` directory, comes in two variants:

- `memoryoptions.bat` for Windows
- `memoryoptions.sh` for Linux

The configuration file, `jvm-memory.options`, is located in the same `bin` directory.

> **Important**
>
> You should not use the `memoryoptions` script when you deploy PingFederate inside of a container. Instead, you should edit the `jvm-memory.options` file directly. In containers, you should use the `InitialRAMPercentage` and `MaxRAMPercentage` JVM options to control the size of the heap.

Installation and upgrade

When the PingFederate installer is executed for Windows or a subsequent rerun of the `memoryoptions` utility, it creates a backup copy of the current `jvm-memory.options` and records the G1GC options in the `jvm-memory.options` file. Changes made as a result of the execution of the utility or a manual edit are activated after a restart of PingFederate.

Depending upon the selected tool and whether the `jvm-memory.options` file exists in the source installation, the expected behavior of the `memoryoptions` utility differs. In general, the `jvm-memory.options` file from the source installation is preserved without new recommended values.

Related links

- memoryoptions and installation
- memoryoptions and upgrade

memoryoptions and installation

When the PingFederate installer for Windows runs the `memoryoptions` utility tool or when changes are made from a manual edit, expect the following behaviors to the installation medium.

The PingFederate installer for Windows runs the `memoryoptions` utility in an attempt to optimize the Java virtual machine (JVM) heap. Regardless of available memory, PingFederate uses the Garbage-First (G1) garbage collector (GC). The script assumes that PingFederate will be the only service on the server and consume a majority of the memory.

The G1GC is designed to achieve high throughput while meeting its pause times goal for garbage collection, and the collector self-tunes by adjusting the size and nature of the various heap regions to meet the pause time goal. As needed, administrators can rerun the utility or manually edit these options at a later time.

When the PingFederate installer is executed for Windows or a subsequent rerun of the `memoryoptions` utility, it creates a backup copy of the current `jvm-memory.options` and records the G1GC options in the `jvm-memory.options` file. Changes made as a result of the execution of the utility or a manual edit are activated after a restart of PingFederate.

> **Important**
>
> You should not use the `memoryoptions` script when you deploy PingFederate inside of a container. Instead, you should edit the `jvm-memory.options` file directly. In containers, you should use the `InitialRAMPercentage` and `MaxRAMPercentage` JVM options to control the size of the heap.

See the following table for information regarding expected behaviors.

PingFederate installation mediums and their expected behaviors from the execution of the **`memoryoptions`** *utility tool*

Installation medium	Expected behavior
PingFederate installer for Windows	• The installer creates a new PingFederate installation. • The installer runs the `memoryoptions` utility. Regardless of available memory, PingFederate uses the G1GC as a default. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. • The installer configures PingFederate to run as a service. • The `memoryoptions` options are activated as the PingFederate service starts.

PingFederate product distribution ZIP file	The default `jvm-memory.options` file becomes part of the new installation as program and default configuration files are extracted from the PingFederate product distribution `.zip` file.
	PingFederate as a console application on Windows or as a console application or a service on Linux
	** The JVM options set in the default `jvm-memory.options` file are activated as PingFederate starts.
	• The default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, run the `memoryoptions` utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. As a result of the execution of the `memoryoptions** utility or a manual edit of the` `jvm-memory.options` `file, the JVM options are activated as PingFederate restarts.`
	`PingFederate as a service on Windows`
	• `When administrators run the PingFederate service-installation program` `install-service.bat`, `located in the` `<pf_install>/pingfederate/sbin/win-x86-64` `directory, to install the PingFederate Windows service manually, the program runs the` `memoryoptions` `utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory.`
	`The service-installation program then runs a helper utility` `generate-wrapper-jvm-options.bat`, `located in the` `<pf_install>/pingfederate/sbin/wrapper` `directory, to read the JVM options from the` `jvm-memory.options` `file and create a resource file that the PingFederate Windows service requires to configure its JVM options`
	• `The default JVM options are activated as the PingFederate service starts.`

memoryoptions and upgrade

Upgrade paths behave differently when changes are executed based on the recommendations and execution of the `memoryoptions` utility tool.

Depending upon the selected tool and whether the `jvm-memory.options` file exists in the source installation, the expected behavior of the `memoryoptions` utility differs. In general, the `jvm-memory.options` file from the source installation is preserved without new recommended values.

> **Important**
>
> You should not use the `memoryoptions` script when you deploy PingFederate inside of a container. Instead, you should edit the `jvm-memory.options` file directly. In containers, you should use the `InitialRAMPercentage` and `MaxRAMPercentage` JVM options to control the size of the heap.

See the following table for information regarding expected behaviors.

PingFederate upgrade paths and their expected behaviors from the execution of the **`memoryoptions`** *utility tool*

Upgrade path	Expected behavior when thejvm-memory.optionsfile does not exist in the source installation
PingFederate installer for Windows	• The installer creates a new PingFederate installation. • The installer runs the `memoryoptions` utility. Regardless of available memory, PingFederate uses the Garbage-First (G1) garbage collector (GC). The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. The script records the defaults in the `jvm-memory.options` file . * The installer configures PingFederate to run as a service. * The recommended options are activated as the PingFederate service starts.

Upgrade path	Expected behavior when thejvm-memory.optionsfile does not exist in the source installation
PingFederate Upgrade Utility (upgrade.bat)	The upgrade utility creates a new PingFederate installation based on the source installation and the PingFederate product distribution `.zip` file. The default `jvm-memory.options` file becomes part of the new installation as the upgrade utility extracts files from the PingFederate product distribution `.zip` file. *PingFederate as a console application on Windows* • The G1GC JVM options set in the default `jvm-memory.options` file are activated as PingFederate starts. • The default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, run the `memoryoptions` utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. As a result of the execution of the memoryoptions** utility or a manual edit of the `jvm-memory.options` file, the JVM options are activated as PingFederate restarts. PingFederate as a service on Windows • When administrators run the PingFederate service-installation program `install-service.bat`, located in the `<pf_install>/pingfederate/sbin/win-x86-64` directory, to install the PingFederate Windows service manually, the program runs the `memoryoptions` utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. The service-installation program then runs a helper utility, `generate-wrapper-jvm-options.bat`, located in the `<pf_install>/pingfederate/sbin/wrapper` directory, to read the JVM options from the `jvm-memory.options` file and create a resource file that the PingFederate Windows service requires to configure its JVM options • The default options are activated as the PingFederate service starts.

Upgrade path	Expected behavior when thejvm-memory.optionsfile does not exist in the source installation
PingFederate Upgrade Utility (upgrade.sh)	* The upgrade utility creates a new PingFederate installation based on the source installation and the PingFederate product distribution `.zip` file. The default `jvm-memory.options` file becomes part of the new installation as the upgrade utility extracts files from the PingFederate product distribution `.zip` file. * The JVM options set in the default `jvm-memory.options` file are activated as PingFederate starts. • The default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, run the **`memoryoptions`** utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. The default options are recorded in the `jvm-memory.options` file. * As a result of the execution of the **`memoryoptions`** utility or a manual edit of the `jvm-memory.options` file, the JVM options are activated as PingFederate restarts.

Upgrade path	Expected behavior when thejvm-memory.optionsfile exists in the source installation
PingFederate installer for Windows	• The installer creates a new PingFederate installation based on the source installation and copies the `jvm-memory.options` file from the source installation to the new installation • At the end of the installation, the installer runs the PingFederate service-installation program, which runs a helper utility `generate-wrapper-jvm-options.bat`, located in the `<pf_install>/pingfederate/sbin/wrapper` directory, to read the JVM options from the `jvm-memory.options` file and create a resource file that the PingFederate Windows service requires to configure its JVM options • The preserved Java virtual machine (JVM) options are activated as the PingFederate service starts.
PingFederate Upgrade Utility (upgrade.sh)	• The upgrade utility creates a new PingFederate installation based on the source installation and copies the `jvm-memory.options` file from the source installation to the new installation • The preserved JVM options are activated as the PingFederate service starts.

Upgrade path	Expected behavior when thejvm-memory.optionsfile exists in the source installation
PingFederate Upgrade Utility (upgrade.bat)	The upgrade utility creates a new PingFederate installation based on the source installation and copies the `jvm-memory.options` file from the source installation to the new installation *PingFederate as a console application on Windows* The preserved JVM options are activated as the PingFederate service starts. *PingFederate as a service on Windows* - When administrators run the PingFederate service-installation program `install-service.bat`, located in the `<pf_install>/pingfederate/sbin/win-x86-64` directory, to install the PingFederate Windows service manually, the program runs the **`memoryoptions`** utility. Regardless of available memory, PingFederate uses the G1GC. The script assumes that PingFederate will be the only service on the server and consume a majority of the memory. The default options are recorded in the `jvm-memory.options` file. The service-installation program then runs a helper utility, `generate-wrapper-jvm-options.bat`, located in the `<pf_install>/pingfederate/sbin/wrapper` directory, to read the JVM options from the `jvm-memory.options` file and create a resource file that the PingFederate Windows service requires to configure its JVM options ** The new recommended options are activated as the PingFederate service starts. **Note** To restore the preserved JVM options from the source installation, see Restoring the preserved JVM.

Restoring the preserved JVM options

When administrators run the PingFederate service-installation program `install-service.bat`, the upgrade utility creates a new PingFederate installation based on the source installation and copies the `jvm-memory.options` file from the source installation to the new installation

About this task

Use the command prompt to edit the `jvm-memory.options` file with a time stamp.

Steps

1. Rename the current `jvm-memory.options` file. For example, `jvm-memory.options.backup`.
2. Look for the preserved `jvm-memory.options` file.

> **Note**
>
> The preserved file was renamed with a time stamp.

3. Remove the time stamp from the file name.

> **Note**
>
> The `jvm-memory.options` is the file preserved from the source installation.

4. Open a command prompt and go to the `<pf_install>/pingfederate/sbin/wrapper` directory.
5. Run `generate-wrapper-jvm-options.bat`.

   > **Note**
   >
   > This helper utility reads the JVM options from the `jvm-memory.options` file and creates a resource file that the PingFederate Windows service requires to configure its JVM options.

6. Close the command prompt.

   The preserved file was renamed with a time stamp.

7. Restart the PingFederate Windows service.

Result

The preserved JVM options are activated as the PingFederate service starts.

Fine-tuning JVM options

Edit the JVM options `jvm-memory.options` file to customize minimum and maximum heap sizing, garbage collection, and generation specific sizing for your memory use and to optimize PingFederate's performance.

About this task

PingFederate reads Java virtual machine (JVM) options from the `jvm-memory.options` file, located in the `<pf_install>/pingfederate/bin` directory. Any manual modifications or additions should be made in this file. For more information on JVM tuning options, see HotSpot Virtual Machine Garbage Collection Tuning Guide in the Oracle documentation.

> **Note**
>
> Before making any edits to the file, consider the following:
>
> - Make a backup copy prior to any manual edits.
> - The empty lines and comments, indicated by a leading `#` character, are ignored.
> - JVM options do not need a specific organization or order.
> - You can add any JVM flag to the file to configure and customize the JVM, not just memory-related options.

Steps

1. Edit the `<pf_install>/pingfederate/bin/jvm-memory.options` file.
2. To add additional JVM options, insert the applicable options to the file.

   Example:

   For example, to enable the aggressive options flag, configure the file as follows.

```
...

# Enable the aggressive options flag
-XX:+AggressiveOpts
```

The comment is optional.

3. When finished, save your changes.
4. If PingFederate is configured to run as a service on a Windows server, follow these steps:
   1. Open command prompt and go to the `<pf_install>/pingfederate/sbin/wrapper` directory.
   2. Run `generate-wrapper-jvm-options.bat`.

      Result:

      This helper utility reads the JVM options from the `jvm-memory.options` file and creates a resource file that the PingFederate Windows service requires to configure its JVM options.
   3. Close the command prompt.
5. Restart PingFederate.
6. For a clustered PingFederate environment, repeat these steps on each engine node as needed.

Hardware security modules

When configuring PingFederate to use a hardware security module, be aware of the following performance impact considerations.

You can configure PingFederate to use a hardware security module (HSM) for cryptographic material storage and operations. When configured, private keys and their corresponding certificate are stored on the HSM. Related signing and decryption operations are processed there for enhanced security. By default, even in HSM mode, dynamic OAuth and OpenID Connect signing and decryption keys are generated and stored in the memory of PingFederate cluster nodes. To ensure continuity after a full cluster restart, the decryption keys are also persisted to disk, and encrypted there with PingFederate’s active configuration encryption key. To ensure OAuth and OpenID Connect keys are instead stored on the HSM, you must enable static keys.

For more information on supported configurations for secure material storing and processing, see Supported hardware security modules.

Performance considerations

Configuring PingFederate to use an HSM for cryptographic material storage and operations can introduce an impact on performance. The level of impact depends on the performance of cryptographic functionality provided by the HSM and the network latency between PingFederate and the HSM. Consult your HSM vendor for performance tuning and optimization recommendations if you plan to use an HSM as part of your PingFederate deployment.

Configuration at scale

You can configure PingFederate to improve the administrative-console experience for your scaling needs.

For deployments that have hundreds of connections or OAuth clients, or both, and observe noticeable delays in the administrative console, administrators can optionally configure PingFederate to create configuration archives during off-peak hours and disable automatic connection validation to improve the administrative-console experience.

Related links

- Configuring a backup schedule
- Configuring automatic connection validation

References

For more information on memory management and Hotspot Java virtual machine (JVM) arguments for garbage collection tuning, see the following resources.

Memory management

For more information, see Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide from Oracle.

Hotspot JVM arguments

For more information, see Java HotSpot VM Options from Oracle.

PingFederate Monitoring Guide

PingFederate provides a range of monitoring options, from simple heartbeat options for checking responsiveness to transaction response-time logging and resource-utilization metrics.

To help you gain insight into the health and performance of your PingFederate deployment, this guide provides the following information:

- Liveliness and responsiveness describes the heartbeat request endpoint and audit logs
- Resource metrics describes mechanisms for obtaining resource metrics including JMX
- Monitoring describes the key JVM performance metrics for evaluating the performance of PingFederate deployments
- Thread pool describes the MBeans tab and the recommended number of threads in the pool
- Logging, reporting, and troubleshooting describes the available logging, reporting, and troubleshooting features

Resource metrics

PingFederate provides mechanisms for obtaining resource metrics including JMX and heartbeat endpoint.

PingFederate provides the following mechanisms for obtaining resource metrics:

- JMX - Ping recommends using JMX MBeans because this method provides a more comprehensive set of resource metric counters for analyzing performance. Several tools are available for collecting and analyzing data from JMX MBeans, including many security information and event management (SIEM) tools like Splunk.
- Heartbeat endpoint - For information about enabling and customizing heartbeat message reporting, see Liveliness and responsiveness.

Monitoring discusses the JConsole monitoring tool that is included with the Java SE platform. For more information about the Comprehensive JConsole, see Troubleshoot with the JConsole Tool in the Oracle JDK documentation and The Java Monitoring and Management Console (jconsole) in the OpenJDK documentation.

Runtime monitoring using JMX

PingFederate supports runtime monitoring and reporting through Java Management Extensions (JMX). JMX technology represents a Java-centric approach to application management and monitoring.

JMX exposes instrumented code in the form of MBeans. Application management systems that support JMX technology, such as JConsole, can request runtime information from the PingFederate JMX server.

> **Important**
>
> Authentication is required for JMX-client access to PingFederate runtime data. For more information, see Configuring service authentication.

> **Tip**
>
> You can use HTTP requests at any time to verify the status of the PingFederate server. For more information, see Customizing the heartbeat message.
> You can also supplement monitoring information by applying third-party analysis and reporting tools to the security audit log, in which PingFederate records fine-grain details, including response times and event types, for all server transactions. For more information, see Security audit logging.

PingFederate JMX server reports monitoring data for single sign-on (SSO) and single logout (SLO) transactions. In addition, numerous Jetty-standard MBeans are available to the PingFederate server's JMX clients.

PingFederate MBeans

PingFederate provides MBeans for tracking server performance. The attributes exposed by these MBeans align with those available at the heartbeat endpoint. Most statistics and counts are calculated over a period. You can configure the period's length in the file `com.pingidentity.monitoring.MonitoringService.xml`.

- `TOTAL_TRANSACTIONS`: The total number of SSO, SLO, and STS transactions processed since the server started. PingFederate resets this counter to zero after restart.
- `TOTAL_FAILED_TRANSACTIONS`: The total number of failed transactions since the server started. PingFederate resets this counter to zero after restart.
- `dataStores`: The request rate and response time statistics for data stores
- `adapters`: The request rate and response time statistics for adapters
- `connections`: The request rate and response time statistics for connections
- `cluster`: Cluster membership and the request rate and response time statistics for cluster Remote Procedure Calls (RPCs)
- `httpRequests.admin`: The request rate and response time statistics for HTTP requests to the administrative console
- `httpRequests.engine`: The request rate and response time statistics for HTTP requests to PingFederate's runtime endpoints
- `stateMaps`: The current number of entries in various runtime state maps
- `transactions`: The counts of all transactions and failed transactions, including values for the previous period and accumulated totals since the server started

Sample Jetty metrics

The following table describes examples of Jetty MBean metrics, available through JMX, that you might find useful to supplement the information that the PingFederate-specific MBeans provide.

MBean	Attributes
`org.eclipse.jetty.io:connectionstatistics` For Jetty connectors including the primary and secondary PingFederate runtime server ports.	`connectionsTotal` – Total number of TCP connections accepted by the server `connectionDuration*` – How long connections are kept open. Maximum, mean, and standard deviation are available `connections` – Current number of open connections. Maximum is also available (`connectionsMax`)
`org.eclipse.jetty.server.handler:statisticshandler`	`requests` – Total number of requests received `requestsActive` – Number of requests currently being processed. Max is also available `requestTime` – Request duration. Maximum, mean, standard deviation, and total accumulated time are available `responses1xx, responses2xx, responses3xx`, ... – Total number of requests that returned HTTP status codes of 1xx, 2xx, 3xx, and so on
`org.eclipse.jetty.util.thread:queuedthreadpool` Two pools: one for the runtime server, with 200 maximum threads; one for the administrative console, with 20 maximum threads.	`idleThreads` – Number of idle threads currently available `threads` – Number of threads currently running, including both idle and active `minThreads` – Minimum number of threads in the pool `maxThreads` – Maximum number of threads in the pool `lowOnThreads` – A boolean flag indicating whether the pool is running low on threads

`java.lang: Memory`
`java.lang: MemoryPool`
`java.lang: GarbageCollection`
`java.lang: OperatingSystem`
	Attributes measuring CPU usage and memory

Advanced JMX configuration

PingFederate uses port 1099 for its JMX server. You can change the port and other Java Message Service (JMS) settings by modifying the `jmx-remote-config.xml` file in the `<pf_install>/pingfederate/server/default/conf` directory.

> **Note**
>
> When connecting to the JMX service using SSL, the default, ensure that the client trusts the PingFederate SSL server certificate presented. For more information, see Manage SSL server certificates.

Connecting with JMX

You can connect to JMX using local and remote processes.

JConsole permits connections to local and remote Java processes. If your instance of PingFederate is running as a Windows Service, you must connect through the remote option. For more information on connecting to a local process, see Connecting to a local process. For information on connecting to a remote process, see Connecting to a remote process.

Connecting to a local process

Unless you are running PingFederate as a Windows service, the easiest method by which to launch JConsole on the same machine as the server is to select Local Process.

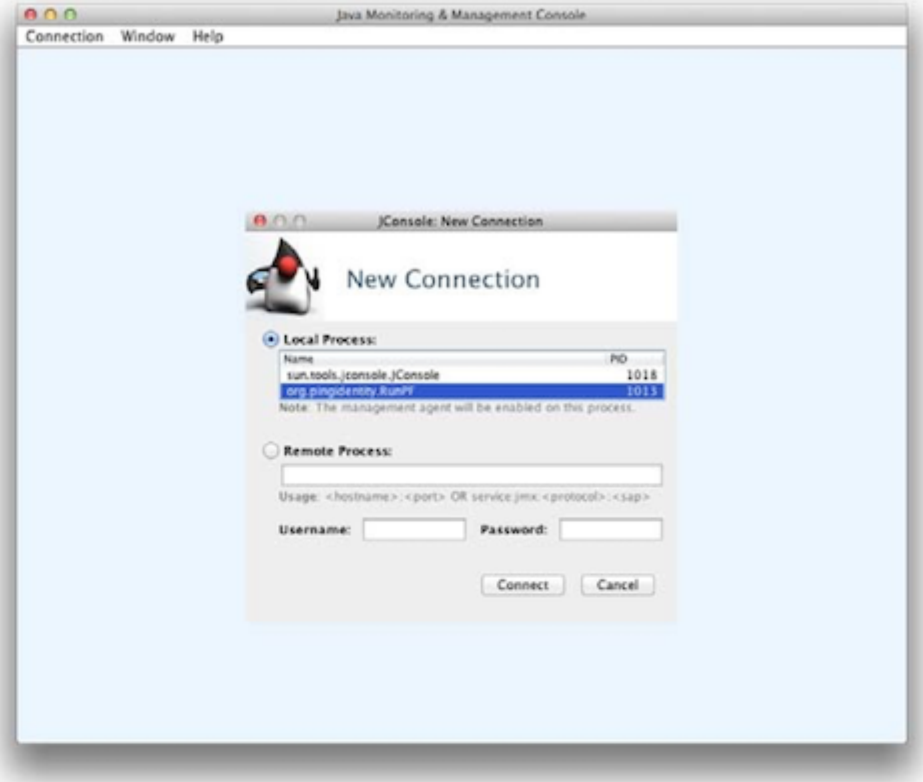
About this task

For information about connecting to a remote process, see Connecting to a remote process.

To connect to a local instance and start the monitoring process:

Steps

- From the Local Process list, select org.pingidentity.RunPF and then click Connect.



● Local Process:

Name	PID
sun.tools.jconsole.JConsole	1018
org.pingidentity.RunPF	1013

Note: The management agent will be enabled on this process.

○ Remote Process:

Usage: <hostname>:<port> OR service:jmx:<protocol>:<sap>

Username: Password:

> **Note**
>
> If you are running the process locally, the system might prompt you to accept the connection as insecure.

Connecting to a remote process

If PingFederate is running as a Windows Service, or if the `org.pingidentity.RunPF` class is unavailable in the Local Process list, use this procedure to establish a connection.

About this task

To enable remote JMX monitoring in PingFederate:

Steps

1. In the Administrative Console, go to the Security > System Integration > Service Authentication window.
2. Define the credentials that are required to connect to the PingFederate JMX service.
3. Restart PingFederate to enable the JMX Service.
4. If PingFederate is deployed in a clustered environment:
    1. Replicate the configuration changes on each node in the cluster.
    2. Restart each engine node.
5. After you enable the JMX service, connect to the remote JMX service by specifying the hostname and port `1099`, or a service URL like the following:

    ```
    service:jmx:rmi:///jndi/rmi://[hostname]:1099/jmxrmi
    ```

    Because JMX uses SSL by default when communicating with a remote host, the client host must trust the PingFederate SSL certificate that is presented when a connection is established.

    To disable the use of SSL for JMX, open the `/server/default/conf/jmx-remote-config.xml` file and set the `<item name="jmx.rmi.ssl">` property to `false`.

    > **Note**
    >
    > If the JMX client does not trust the JMX certificate, a `connection failed` SSL message appears.

6. If SSL is enabled in `jmx-remote-config.xml`, import the PingFederate SSL certificate to the client's trusted certificates.
7. If SSL is disabled, click Insecure to connect.

Liveliness and responsiveness

One of the simpler methods for monitoring the performance of a PingFederate deployment involves determining whether the PingFederate Server is available and responsive. To help you identify the status of a server, PingFederate provides a heartbeat request endpoint.

Heartbeat endpoint

If the PingFederate server is running, the process of sending a request to the endpoint `/pf/heartbeat.ping` returns an `HTTP 200` status. If the request times out or requires an extended amount of time to return, the server might be overloaded or experiencing other difficulties.

If a request requires more than two or three seconds to return, multiple factors in your PingFederate deployment might be responsible. We recommend that you develop a baseline for the desired response time by testing the heartbeat endpoint of your deployment at various times. This endpoint can be useful when load balancing a cluster of PingFederate instances. Some load balancers can alter the number of requests that are sent to a particular server based on the response code received, or the responsiveness of requests that are made to the heartbeat endpoint.

The output of the heartbeat endpoint can be modified to provide performance-related information, such as CPU and memory usage, and response times. The response metrics can help you make better auto-scaling decisions. The map size metrics can help you recognize performance issues.

The following example shows a report containing a sample of the PingFederate server metrics available from the heartbeat endpoint. This response does not include examples of every field. For a complete list of fields, see the server metrics table below.

```
{"items":[{
"cpu.load": "14.86",
"total.jvm.memory": "536.871 MB",
"free.jvm.memory": "305.31 MB",
"used.jvm.memory": "231.561 MB",
"total.physical.system.memory": "68.719 GB",
"total.free.physical.system.memory": "30789.845 MB",
"total.used.physical.system.memory": "37.93 GB",
"number.of.cpus": "12",
"response.statistics.count": "540",
"response.statistics.window.seconds": "15",
"response.time.statistics.90.percentile": "159.350784",
"response.time.statistics.max": "6343.0",
"response.time.statistics.mean": "23.522222222222222",
"response.time.statistics.min": "0.0",
"response.concurrency.statistics.90.percentile": "2.0625",
"response.concurrency.statistics.max": "2.0",
"response.concurrency.statistics.mean": "0.7222222222222222",
"response.concurrency.statistics.min": "0.0",
"response.http.status.1xx": "0",
"response.http.status.2xx": "360",
"response.http.status.3xx": "180",
"response.http.status.4xx": "0",
"response.http.status.5xx": "0",
"transaction.count": "240",
"transaction.errors": "0",
"total.transactions": "660",
"total.failed.transactions": "0",
"ds.JDBC.PFIndexDS.request.count": "360",
"ds.JDBC.PFIndexDS.response.time.90.percentile": "0.0",
"ds.JDBC.PFIndexDS.response.time.max": "3.0",
"ds.JDBC.PFIndexDS.response.time.mean": "0.030555555555555555",
"ds.JDBC.PFIndexDS.response.time.min": "0.0",
"ds.LDAP.LDAP-8C4A5F60684C90B9ECE388D2B7194F7909C804CF.request.count": "485",
"ds.LDAP.LDAP-8C4A5F60684C90B9ECE388D2B7194F7909C804CF.response.time.90.percentile": "50.29888",
"ds.LDAP.LDAP-8C4A5F60684C90B9ECE388D2B7194F7909C804CF.response.time.max": "5964.0",
"ds.LDAP.LDAP-8C4A5F60684C90B9ECE388D2B7194F7909C804CF.response.time.mean": "21.25773195876289",
"ds.LDAP.LDAP-8C4A5F60684C90B9ECE388D2B7194F7909C804CF.response.time.min": "0.0",
"adapter.CIAMHtml.lookupAuthN.90.percentile": "100.630528",
"adapter.CIAMHtml.lookupAuthN.count": "121",
"adapter.CIAMHtml.lookupAuthN.max": "6123.0",
"adapter.CIAMHtml.lookupAuthN.mean": "43.768595041322314",
"adapter.CIAMHtml.lookupAuthN.min": "0.98304",
"connection.https://pfdev.ping-eng.com:9031.jwks.90.percentile": "0.0",
"connection.https://pfdev.ping-eng.com:9031.jwks.count": "0",
"connection.https://pfdev.ping-eng.com:9031.jwks.max": "46.0",
"connection.https://pfdev.ping-eng.com:9031.jwks.mean": "0.0",
"connection.https://pfdev.ping-eng.com:9031.jwks.min": "0.0",
"connection.https://pfdev.ping-eng.com:9031.token.90.percentile": "3.93216",
"connection.https://pfdev.ping-eng.com:9031.token.count": "60",
"connection.https://pfdev.ping-eng.com:9031.token.max": "1044.0",
"connection.https://pfdev.ping-eng.com:9031.token.mean": "3.9",
"connection.https://pfdev.ping-eng.com:9031.token.min": "2.883584",
"connection.https://pfdev.ping-eng.com:9031.userinfo.90.percentile": "2.981888",
"connection.https://pfdev.ping-eng.com:9031.userinfo.count": "60",
"connection.https://pfdev.ping-eng.com:9031.userinfo.max": "18.0",
"connection.https://pfdev.ping-eng.com:9031.userinfo.mean": "2.4166666666666665",
"connection.https://pfdev.ping-eng.com:9031.userinfo.min": "0.98304",
"engine.jetty.queued.thread.pool.max.available.threads": "199",
"engine.jetty.queued.thread.pool.queue.size": "0",
"engine.jetty.queued.thread.pool.utilization.rate": "0.01507537688442211",
"engine.jetty.queued.thread.pool.utilized.threads": "3",
"idp.session.registry.session.map.size": "165",
"sp.session.registry.session.map.size": "165",
"session.state.attribute.map.size": "166",
"transaction.state.map.size": "1",
"atm.default.token.map.size": "0",
"cluster.members": "[172.31.28.63:7600, 172.31.29.114:7600]",
"cluster.rpc.addKeys.90.percentile": "0.0",
"cluster.rpc.addKeys.count": "0",
"cluster.rpc.addKeys.max": "0.0",
"cluster.rpc.addKeys.mean": "0.0",
"cluster.rpc.addKeys.min": "0.0",
"cluster.rpc.getAttr.90.percentile": "0.98304",
"cluster.rpc.getAttr.count": "423",
"cluster.rpc.getAttr.max": "1.0",
"cluster.rpc.getAttr.mean": "0.1276595744680851",
"cluster.rpc.getAttr.min": "0.0",
"cluster.rpc.getAuthnSessionInfo.90.percentile": "0.98304",
"cluster.rpc.getAuthnSessionInfo.count": "121",
"cluster.rpc.getAuthnSessionInfo.max": "4.0",
"cluster.rpc.getAuthnSessionInfo.mean": "0.371900826446281",
"cluster.rpc.getAuthnSessionInfo.min": "0.0",
"cluster.rpc.registerBeans.90.percentile": "0.98304",
"cluster.rpc.registerBeans.count": "121",
"cluster.rpc.registerBeans.max": "9.0",
"cluster.rpc.registerBeans.mean": "0.45454545454545453",
"cluster.rpc.registerBeans.min": "0.0",
"cluster.rpc.registerSriToUniqueUserKey.90.percentile": "0.98304",
"cluster.rpc.registerSriToUniqueUserKey.count": "122",
"cluster.rpc.registerSriToUniqueUserKey.max": "1.0",
"cluster.rpc.registerSriToUniqueUserKey.mean": "0.1885245901639344",
"cluster.rpc.registerSriToUniqueUserKey.min": "0.0",
"cluster.rpc.removeAttr.90.percentile": "0.98304",
"cluster.rpc.removeAttr.count": "361",
"cluster.rpc.removeAttr.max": "1.0",
"cluster.rpc.removeAttr.mean": "0.16620498614958448",
"cluster.rpc.removeAttr.min": "0.0",
"cluster.rpc.retrieveAndRemoveState.90.percentile": "0.98304",
"cluster.rpc.retrieveAndRemoveState.count": "180",
"cluster.rpc.retrieveAndRemoveState.max": "3.0",
"cluster.rpc.retrieveAndRemoveState.mean": "0.47222222222222227",
"cluster.rpc.retrieveAndRemoveState.min": "0.0",
"cluster.rpc.saveState.90.percentile": "0.98304",
"cluster.rpc.saveState.count": "180",
"cluster.rpc.saveState.max": "10.0",
"cluster.rpc.saveState.mean": "0.55",
"cluster.rpc.saveState.min": "0.0",
"cluster.rpc.setAttr.90.percentile": "0.98304",
"cluster.rpc.setAttr.count": "301",
"cluster.rpc.setAttr.max": "20.0",
"cluster.rpc.setAttr.mean": "0.318936877076412",
"cluster.rpc.setAttr.min": "0.0",
"cluster.rpc.synchronizeKeys.90.percentile": "2.883584",
"cluster.rpc.synchronizeKeys.count": "0",
"cluster.rpc.synchronizeKeys.max": "3.0",
"cluster.rpc.synchronizeKeys.mean": "0.0",
"cluster.rpc.synchronizeKeys.min": "2.883584"
}]}
```

The following table describes all the PingFederate server metrics available from the heartbeat endpoint.

> **Note**
>
> In the following table, for server metrics that end in `.90.percentile`, the current `90` value is determined by the `ServerPercentilesList` item in the `com.pingidentity.monitoring.MonitoringService.xml` file. `90` is the default value. For more information on how to edit this value, see step 4 in Customizing the heartbeat message.

Server metrics	Description
`cpu.load`	Load on the PingFederate server's cores as a percentage of total capacity
`total.jvm.memory`	Total memory of the JVM
`free.jvm.memory`	Free memory of the JVM
`used.jvm.memory`	Used memory of the JVM
`total.physical.system.memory`	Total system memory
`total.free.physical.system.memory`	Free system memory
`total.used.physical.system.memory`	Used system memory
`number.of.cpus`	Number of cores on the PingFederate server
`response.statistics.count`	Number of items considered in the heartbeat report for the time and concurrency statistics
`response.statistics.window.seconds`	Time interval (in seconds) for the statistics report (this is an echo of the `StatisticsWindowSecs` parameter's value and provides context for the concurrency and time statistics)
`response.time.statistics.90.percentile`	The 90th percentile response time in milliseconds during the statistics window (for example, if this value is 168, then 90% of the report samples had response times below 168 milliseconds)
`response.time.statistics.max`	Longest time in milliseconds that the PingFederate server took to respond during the statistics window
`response.time.statistics.mean`	Mean time in milliseconds that the PingFederate server took to respond during the statistics window
`response.time.statistics.min`	Shortest time in milliseconds that the PingFederate server took to respond during the statistics window

Server metrics	Description
`response.concurrency.statistics.90.percentile`	The 90th percentile response concurrency during the statistics window (for example, if this value is 124, then 90% of the report samples had response concurrency values below 124)
`response.concurrency.statistics.max`	Maximum number of HTTP requests that the PingFederate server processed concurrently during the statistics window
`response.concurrency.statistics.mean`	Mean number of HTTP requests that the PingFederate server processed concurrently during the statistics window
`response.concurrency.statistics.min`	Minimum number of HTTP requests that the PingFederate server processed concurrently during the statistics window
`response.http.status.1xx`	Number of 1xx HTTP response codes during the statistics window
`response.http.status.2xx`	Number of 2xx HTTP response codes during the statistics window
`response.http.status.3xx`	Number of 3xx HTTP response codes during the statistics window
`response.http.status.4xx`	Number of 4xx HTTP response codes during the statistics window
`response.http.status.5xx`	Number of 5xx HTTP response codes during the statistics window
`transaction.count`	Number of SSO, SLO, and STS transactions during the statistics window
`transaction.errors`	Number of failed SSO, SLO, and STS transactions during the statistics window
`total.transactions`	Total number of SSO, SLO, and STS transactions since the server started
`total.failed.transactions`	Total number of failed SSO, SLO, and STS transactions since the server started
`ds.<type>.<id>.request.count`	Number of requests for the data store during the statistics window
`ds.<type>.<id>.response.time.90.percentile`	The data store's 90th percentile response time in milliseconds during the statistics window
`ds.<type>.<id>.response.time.mean`	The data store's mean response time in milliseconds during the statistics window
`ds.<type>.<id>.response.time.min`	The data store's minimum response time in milliseconds during the statistics window
`ds.<type>.<id>.response.time.max`	The data store's maximum response time in milliseconds during the statistics window
`ds.<type>.<id>.errors`	Number of data store errors during the statistics window

Server metrics	Description
`adapter.<adapter id>.lookupAuthN.count`	Number of authentication requests for the adapter during the statistics window
`adapter.<adapter id>.lookupAuthN.90.percentile`	The adapter's 90th percentile response time in milliseconds during the statistics window
`adapter.<adapter id>.lookupAuthN.mean`	The adapter's mean response time in milliseconds during the statistics window
`adapter.<adapter id>.lookupAuthN.min`	The adapter's minimum response time in milliseconds during the statistics window
`adapter.<adapter id>.lookupAuthN.max`	The adapter's maximum response time in milliseconds during the statistics window
`adapter.<adapter id>.lookupAuthN.errors`	Number of failed adapter authentication requests during the statistics window
`connection.<issuer id>.jwks.count`	Number of requests for the OIDC identity provider (IdP) connection JWKS endpoint during the statistics window
`connection.<issuer id>.jwks.90.percentile`	The OIDC IdP connection JWKS endpoint's 90th percentile response time in milliseconds during the statistics window
`connection.<issuer id>.jwks.mean`	The OIDC IdP connection JWKS endpoint's mean response time in milliseconds during the statistics window
`connection.<issuer id>.jwks.min`	The OIDC IdP connection JWKS endpoint's minimum response time in milliseconds during the statistics window
`connection.<issuer id>.jwks.max`	The OIDC IdP connection JWKS endpoint's maximum response time in milliseconds during the statistics window
`connection.<issuer id>.jwks.errors`	Number of failed OIDC IdP connection JWKS endpoint requests during the statistics window
`connection.<issuer id>.token.count`	Number of requests for the OIDC IdP connection token endpoint during the statistics window
`connection.<issuer id>.token.90.percentile`	The OIDC IdP connection token endpoint's 90th percentile response time in milliseconds during the statistics window
`connection.<issuer id>.token.mean`	The OIDC IdP connection token endpoint's mean response time in milliseconds during the statistics window
`connection.<issuer id>.token.min`	The OIDC IdP connection token endpoint's minimum response time in milliseconds during the statistics window

Server metrics	Description
`connection.<issuer id>.token.max`	The OIDC IdP connection token endpoint's maximum response time in milliseconds during the statistics window
`connection.<issuer id>.token.errors`	Number of failed OIDC IdP connection token endpoint requests during the statistics window
`connection.<issuer id>.userinfo.count`	Number of requests for the OIDC IdP connection user info endpoint during the statistics window
`connection.<issuer id>.userinfo.90.percentile`	The OIDC IdP connection user info endpoint's 90th percentile response time in milliseconds during the statistics window
`connection.<issuer id>.userinfo.mean`	The OIDC IdP connection user info endpoint's mean response time in milliseconds during the statistics window
`connection.<issuer id>.userinfo.min`	The OIDC IdP connection user info endpoint's minimum response time in milliseconds during the statistics window
`connection.<issuer id>.userinfo.max`	The OIDC IdP connection user info endpoint's maximum response time in milliseconds during the statistics window
`connection.<issuer id>.userinfo.errors`	Number of failed OIDC IdP connection user info endpoint requests during the statistics window
`connection.<entity id>.ars.count`	Number of requests for the SAML IdP connection artifact endpoint during the statistics window
`connection.<entity id>.ars.90.percentile`	The SAML IdP connection artifact endpoint's 90th percentile response time in milliseconds during the statistics window
`connection.<entity id>.ars.mean`	The SAML IdP connection artifact endpoint's mean response time in milliseconds during the statistics window
`connection.<entity id>.ars.min`	The SAML IdP connection artifact endpoint's minimum response time in milliseconds during the statistics window
`connection.<entity id>.ars.max`	The SAML IdP connection artifact endpoint's maximum response time in milliseconds during the statistics window
`connection.<entity id>.ars.errors`	Number of failed SAML IdP connection artifact endpoint requests during the statistics window
`<admin|engine>.jetty.queued.thread.pool.utilized.threads`	Number of threads in the Jetty thread pool that are currently in use

Server metrics	Description
`<admin|engine>.jetty.queued.thread.pool.max.available.threads`	Maximum number of threads in the Jetty thread pool
`<admin|engine>.jetty.queued.thread.pool.utilization.rate`	The threads in the pool that are currently in use, as a fraction of the maximum available threads
`<admin|engine>.jetty.queued.thread.pool.queue.size`	Number of requests currently queued waiting to be handled by a thread in the pool
`idp.session.registry.session.map.size`	Number of IdP sessions
`idp.session.registry.session.map.purge.unexpired`	Number of unexpired entries purged from the IdP session registry during the statistics window
`sp.session.registry.session.map.size`	Number of service provider (SP) sessions
`sp.session.registry.session.map.purge.unexpired`	Number of unexpired entries purged from the SP session registry during the statistics window
`session.state.attribute.map.size`	Number of items in the session state attribute map
`session.state.attribute.map.purge.unexpired`	Number of unexpired entries purged from the session state attribute map during the statistics window
`transaction.state.map.size`	Number of items in the SSO transaction state map
`transaction.state.map.purge.unexpired`	Number of unexpired entries purged from the SSO transaction state map during the statistics window
`atm.<atm>.token.map.size`	Number of tokens in the access token manager with the ID specified by *<atm>*
`cluster.members`	Holds the cluster membership list
`cluster.rpc.<rpc name>.90.percentile`	The synchronous cluster Remote Procedure Call (RPC)'s 90th percentile response time in milliseconds during the statistics window
`cluster.rpc.<rpc name>.mean`	The synchronous cluster RPC's mean response time in milliseconds during the statistics window

Server metrics	Description
`cluster.rpc.<rpc name>.min`	The synchronous cluster RPC's minimum response time in milliseconds during the statistics window
`cluster.rpc.<rpc name>.max`	The synchronous cluster RPC's maximum response time in milliseconds during the statistics window
`cluster.rpc.<rpc name>.errors`	Number of cases where the RPC received no valid responses

As indicated in the table, the values of some metrics are calculated over a configurable time window. The default statistics window is five minutes.

To customize the statistics window period, change the value of the `StatisticsWindowSecs` parameter in the `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.monitoring.MonitoringService.xml file`. This file also lets you specify additional JMX MBean attributes that will be made available to the heartbeat page templates.

For more information, see Customizing the heartbeat message.

Response-time logging

By default, the audit logs record the processing time for each transaction. With audit logging enabled, you can identify the speed with which PingFederate processes the following transaction types:

- Single sign-on (SSO)
- OAuth
- Security token services (STS)

Depending on your logging configuration, audit logging might not log any transactions. For more information, see Security audit logging.

The following provides examples of the default audit log.

```
2019-11-10 13:24:57,493| tid:cYunBsgybiw_fiRnJjkAhbIXvzc| AUTHN_SESSION_USED| | 127.0.0.1 | | ac_client| |
localhost| IdP| success| PdFormAdpt| |  17
```

```
2019-11-10 13:24:58,720| tid:cYunBsgybiw_fiRnJjkAhbIXvzc| OAuth| 5c60f022-1e9d-3fbe-9749-4b9ca5591356|
127.0.0.1 | | ac_client| OAuth20| localhost| AS| success| PdFormAdpt| |  7
```

Processing times are shown at the end of the entry in milliseconds.

Monitoring

This topic outlines the key JVM performance metrics for evaluating the performance of a PingFederate deployment.

After a connection is established, you can access the JConsole monitoring interface.

Monitoring clustered PingFederate engines

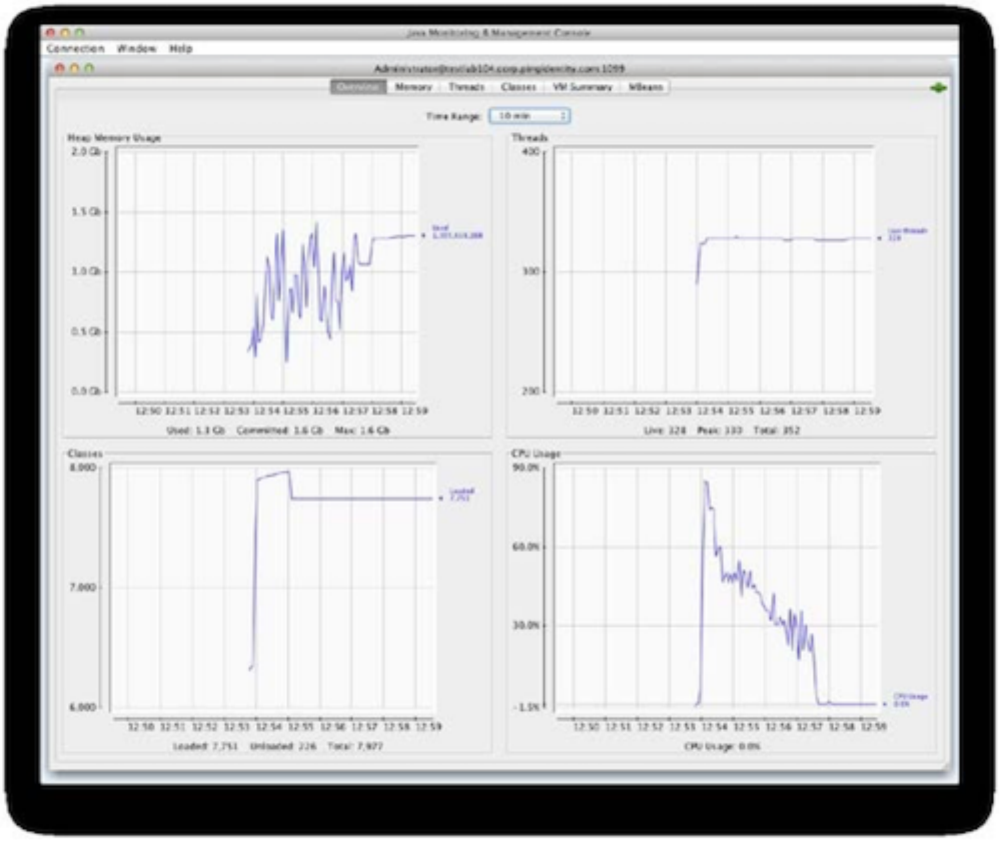
JConsole can be connected to multiple processes. To monitor several instances of PingFederate after a connection is established, click Connection > New Connection and add the additional connection.

Monitoring CPU utilization

The Overview tab provides a dashboard of the following performance and resource-utilization charts:

- Heap Memory Usage (cumulative memory that is used by all memory pools).
- Live Threads
- CPU Usage
- Classes (number of classes that are loaded)

This tab provides a high-level view of the JVM’s performance metrics.



Time Range: 10 min

Used: 1.3 Gb Committed: 1.6 Gb Max: 1.6 Gb

Live: 328 Peak: 333 Total: 352

Loaded: 7,751 Unloaded: 226 Total: 7,977

CPU Usage: 0.0%

Use the Overview tab to visualize and collect CPU usage data. When your PingFederate deployment is subjected to its normal or expected load, the CPU utilization typically falls between 60 and 80%. If the system registers consistently at 80% or higher, additional CPU resources might be necessary to handle load spikes that occur during peak usage times.

Monitoring memory utilization

The Overview tab shows only overall heap usage. To view additional details about memory utilization, click the Memory tab, which lets you analyze usage patterns in specific memory pools within the heap. This tab also provides information about the overall heap utilization profile.

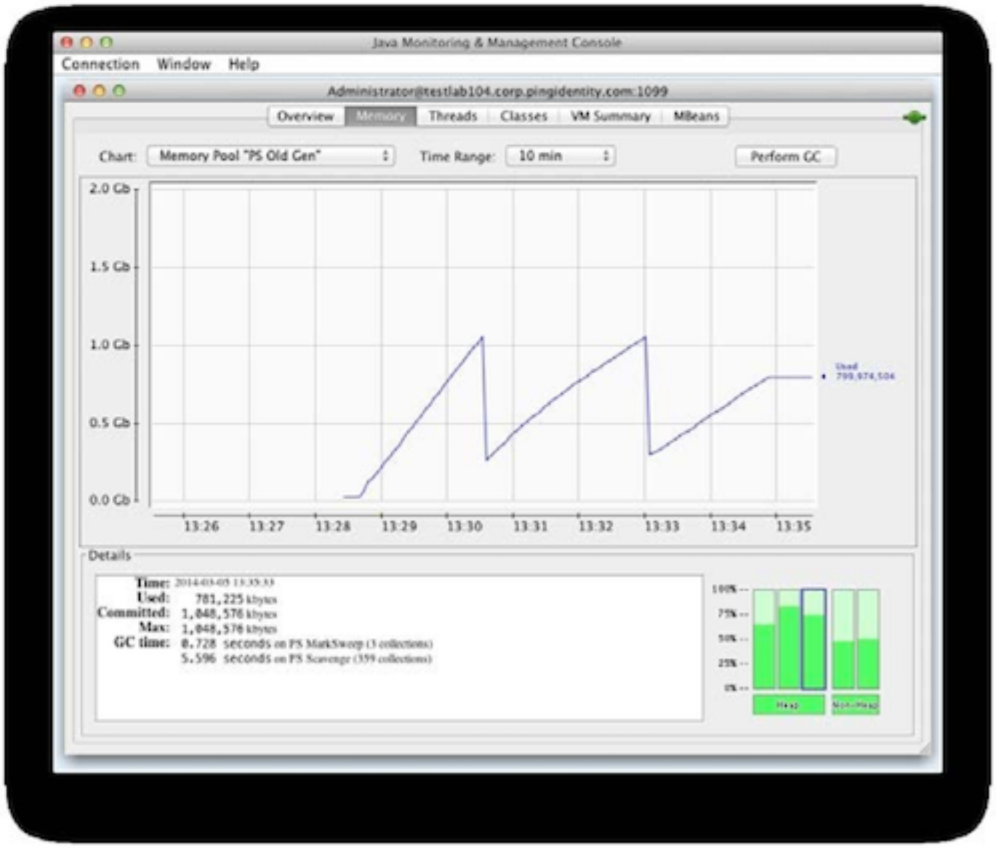
Old Generation space

Objects that survive a sufficient number of garbage-collection cycles are promoted to the Old Generation. To view the memory usage in the pool of such objects, click Memory Pool > PS Old Gen or Memory Pool > G1 Old, depending on the relevant garbage collection. PingFederate services mostly short-lived transactions, like SSO, STS, and OAuth requests and most of the created memory objects are required only for a short period of time.

Although PingFederate makes use of some memory objects that are medium to long lived, such as session data for authentication session, adapter sessions, or single logout functionality, most of the objects that are promoted to the Old Generation are likely to become garbage that requires cleaning up. If the younger generation, or *Eden space*, is not sized appropriately, objects are moved to and retained in the Old Generation before they are collected as garbage. If size limitations prevent the Old Generation from accumulating future garbage as well as longer-lived objects, then garbage-collection cycles occur more frequently.

The Old Generation space is the most important space to monitor. It is easy to identify if the heap is sized and proportioned appropriately for a specific load, based on its usage pattern. The following examples involve two Old Generation usage charts. In both examples, the following examples involve two Old Generation usage charts. In both examples, the same user load executes the same workflow. The size of the heap represents the only difference.

Garbage collection frees around 60 to 75% of the space, and room is available to accommodate the future garbage of newly created objects that are moved from the Eden space, as well as the longer-term objects that remain in use. Although the space is 1 GB in size, the average full (PS MarkSweep or G1 Old Generation) collection time is approximately only 240 milliseconds (0.728 seconds for three collections).



```
          Time: 2014-03-05 13:35:33
          Used:    781,225 kbytes
     Committed: 1,048,576 kbytes
           Max: 1,048,576 kbytes
      GC time: 0.728 seconds on PS MarkSweep (3 collections)
               5.596 seconds on PS Scavenge (359 collections)
```

When a heap is sized inadequately, the Old Generation runs out of space. In the following example, the amount of memory that becomes free with each garbage collection shrinks, due to the rate at which objects are promoted from the Eden space.

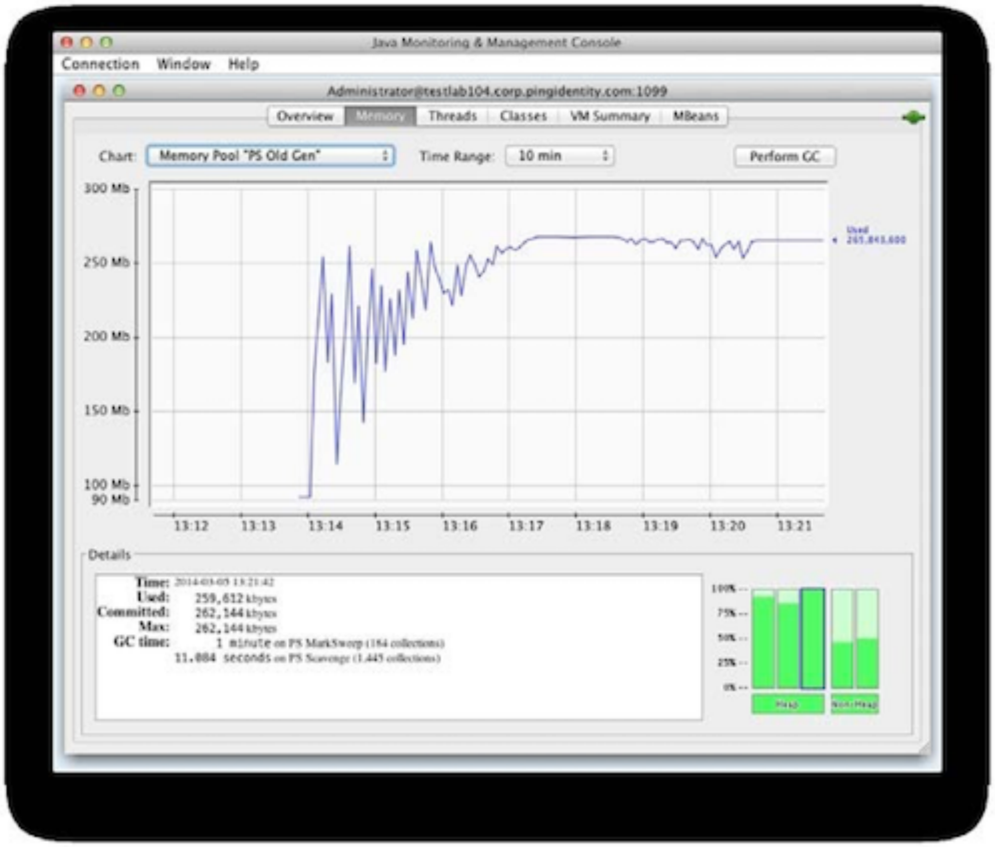


Chart: Memory Pool "PS Old Gen"

Time Range: 10 min

Details

```
       Time: 2014-03-05 13:21:42
       Used:    259,612 kbytes
  Committed:    262,144 kbytes
        Max:    262,144 kbytes
    GC time:          1 minute on PS MarkSweep (184 collections)
             11.084 seconds on PS Scavenge (1,445 collections)
```

184 PS MarkSweep (full) collections require garbage collections more frequently, totaling 60 seconds, or an average of 326 milliseconds per collection.

Entire heap space

If the heap is sized appropriately for the load that the system must handle, it fills up and is followed by an appreciable drop in usage as a full garbage collection occurs (such as a PS MarkSweep collection triggered by the Old Generation filling up). In this example, the heap rises steadily, with drops from minor collections until a PS MarkSweep collection occurs and collects approximately 70% of the heap.

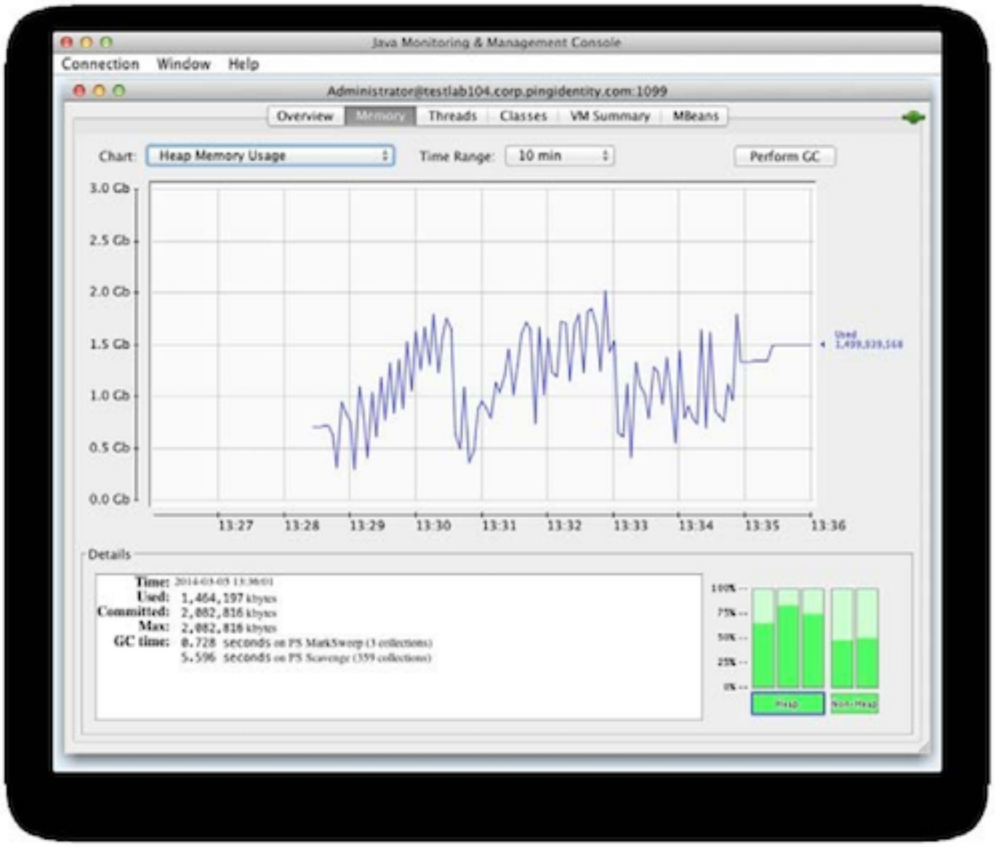
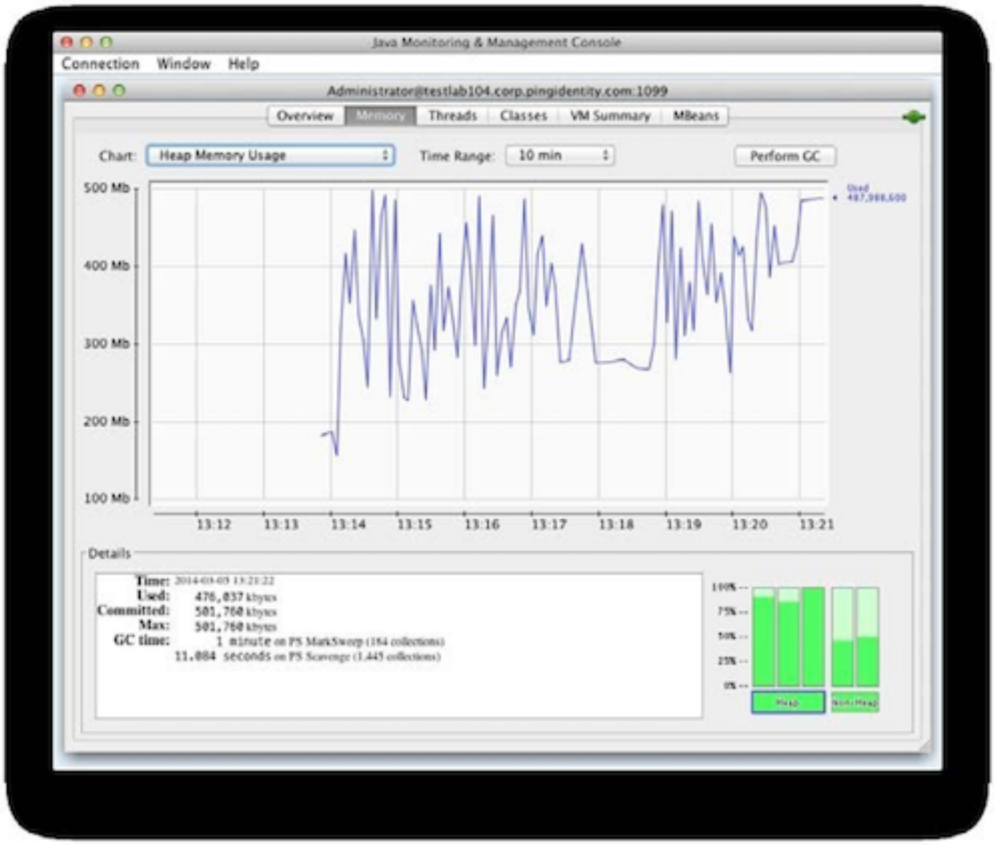


Chart: Heap Memory Usage
Time Range: 10 min

Details

```
      Time: 2014-03-05 13:36:01
      Used: 1,464,197 kbytes
 Committed: 2,082,816 kbytes
       Max: 2,082,816 kbytes
   GC time: 0.728  seconds on PS MarkSweep (3 collections)
            5.596  seconds on PS Scavenge (359 collections)
```

When the heap is undersized, full collections that are performed more frequently return less memory. In the following example, the frequency of JMX data that the JConsole retrieves does not keep pace with the frequency of full collections. As a result, only a fraction of them occur.



```
      Time: 2014-03-05 13:21:22
      Used:    476,037 kbytes
 Committed:    501,760 kbytes
       Max:    501,760 kbytes
  GC time:         1 minute on PS MarkSweep (184 collections)
             11.084 seconds on PS Scavenge (1,445 collections)
```

Eden space

Regardless of whether the heap is adequately sized or undersized, the usage pattern is nearly identical with the Eden space. This similarity can be due to the sampling frequency of the data-collection tool because the number of samples might be insufficient to show that, with an undersized heap, memory is consumed and subsequently freed with greater frequency. The behavior of garbage collection in the Eden space is such that when it fills, the space is completely emptied by moving live objects to the Survivor and Old Generation spaces. Under load, the pattern resembles a jagged sawtooth, as shown in the following examples of an adequately sized heap and an undersized heap.

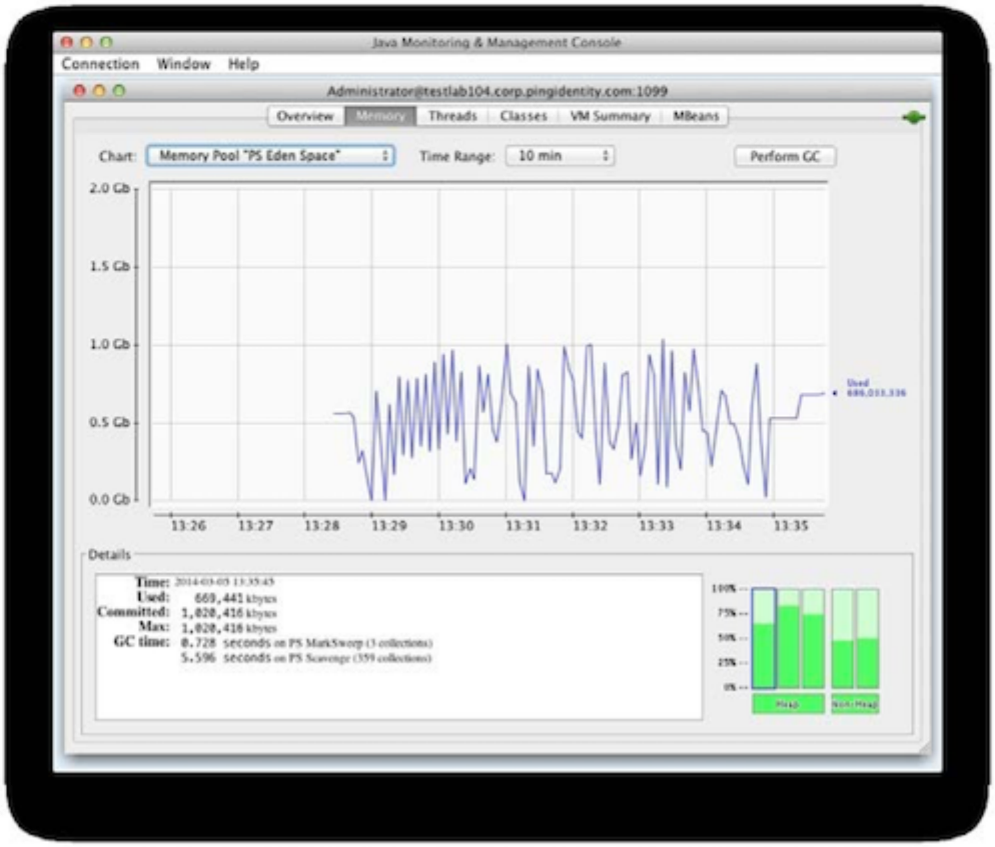


Chart: Memory Pool "PS Eden Space" Time Range: 10 min

```
     Time: 2014-03-05 13:35:45
     Used:    669,441 kbytes
Committed:  1,020,416 kbytes
      Max:  1,020,416 kbytes
  GC time:  0.728  seconds on PS MarkSweep (3 collections)
            5.596  seconds on PS Scavenge (359 collections)
```

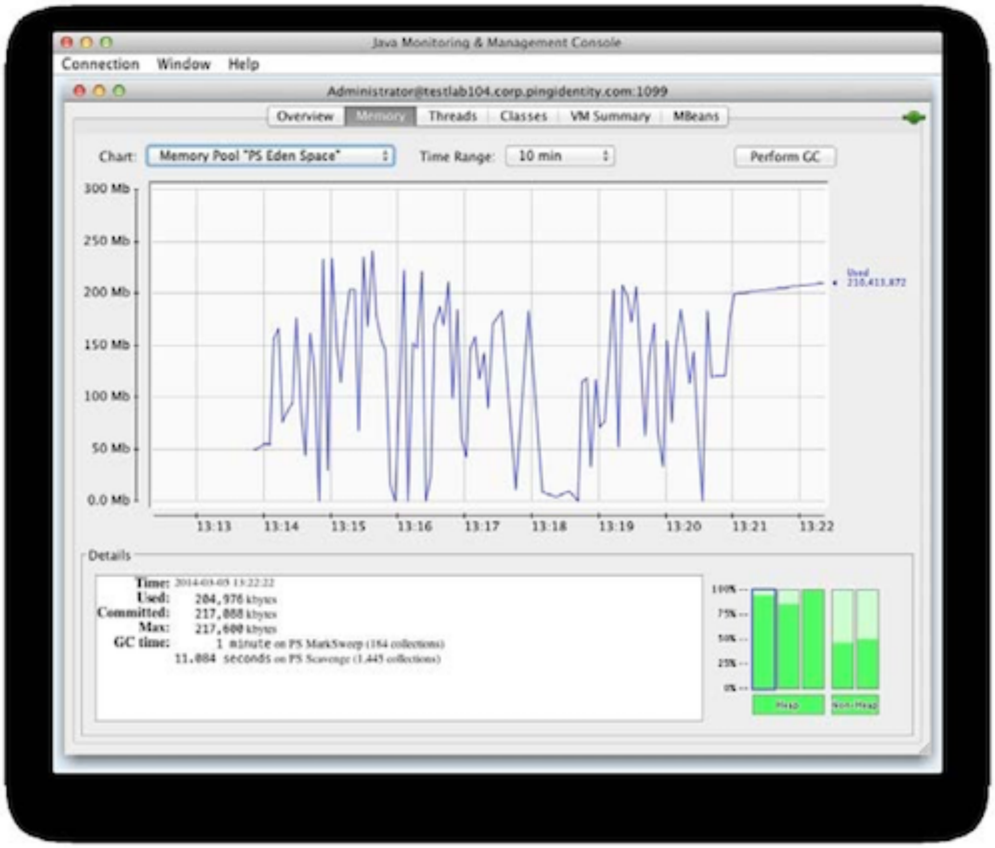


Chart: Memory Pool "PS Eden Space" Time Range: 10 min

```
     Time: 2014-03-05 13:22:22
     Used:    204,976 kbytes
Committed:    217,088 kbytes
      Max:    217,600 kbytes
  GC time:        1  minute on PS MarkSweep (184 collections)
           11.084  seconds on PS Scavenge (1,445 collections)
```

Increasing heap size

Because garbage collectors manage memory in the Java Runtime Environment, simply increasing the size of the heap is not always the appropriate solution. The following table outlines the total heap size recommendations for the available garbage collectors, based on available CPU resources.

Garbage Collector	Minimum Recommended Number of CPUs	Recommended Heap Size
Parallel	4	6 GB maximum
Concurrent Mark Sweep	12	4 - 6 GB minimum
Garbage First (G1)	12	6 GB minimum

If additional memory is unavailable, or if increasing the size of the heap is inadvisable because of these recommendations, the load that is handled by this instance is probably too high. In such instances, consider adding additional resources to your deployment. To verify whether the load for the instance is too high, check the CPU utilization

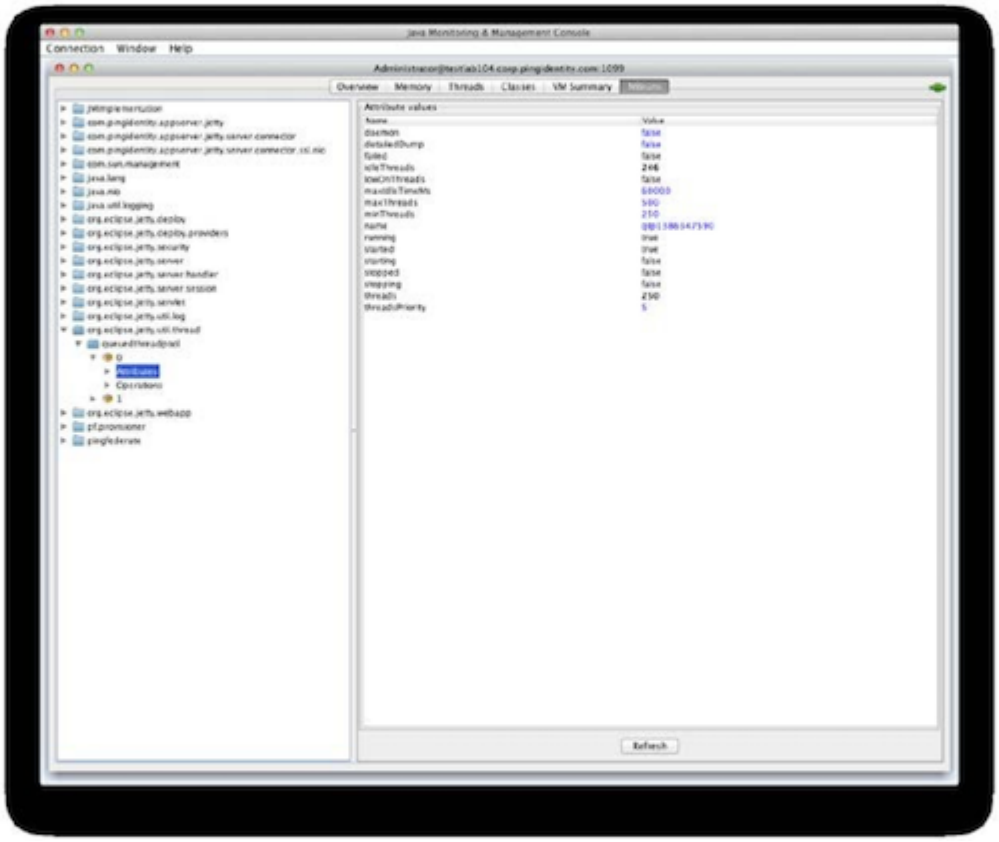
To allow for the most efficient management of memory, set the minimum and maximum heap sizes to the maximum allowed values to avoid potentially expensive heap allocation resizing and divide it evenly between the young and old generations. If you are using the Garbage First collector, generational spaces are not specified through command line options because they are managed logically in real time. Even in such instances, we recommend setting the minimum and maximum heap sizes to the maximum allowed values.

Learn more about fine-tuning the JVM options in the `jvm-memory.options` file in Fine-tuning JVM options.

Thread pool

The following topic describes the MBeans tab and the recommended number of threads in the pool.

The MBeans tab provides access to the JMX Managed Beans and their available attributes and operations. Of particular interest are the `queuedthreadpool` instances that are available within the `org.eclipse.jetty.util.thread` bean. For example, instance 0 represents the thread pool that handles runtime requests. When you click the Attributes item for instance 0, the current state of the thread pool is displayed.


Java Monitoring & Management Console
Connection Window Help
Administrator@testlab104.corp.pingidentity.com:1099
Overview Memory Threads Classes VM Summary MBeans
jmimplementation
com.pingidentity.appserver.jetty
com.pingidentity.appserver.jetty.server.connector
com.pingidentity.appserver.jetty.server.connector.ssl.nio
com.sun.management
java.lang
java.nio
java.util.logging
org.eclipse.jetty.deploy
org.eclipse.jetty.deploy.providers
org.eclipse.jetty.security
org.eclipse.jetty.server
org.eclipse.jetty.server.handler
org.eclipse.jetty.server.session
org.eclipse.jetty.servlet
org.eclipse.jetty.util.log
org.eclipse.jetty.util.thread
queuedthreadpool
0
Attributes
Operations
1
org.eclipse.jetty.webapp
pf.provisioner
pingfederate
Attribute values
Refresh

Name	Value
daemon	false
detailedDump	false
failed	false
idleThreads	246
lowOnThreads	false
maxIdleTimeMs	60000
maxThreads	500
minThreads	250
name	qtp1386547590
running	true
started	true
starting	false
stopped	false
stopping	false
threads	250
threadsPriority	5

The number of threads in the pool (the `threads` attribute) can be compared to the number of threads that are not currently in use (the `idleThreads` attribute). Ideally, a sufficient number of threads is available to handle load spikes that occur during peak usage times, while also limiting the number of idle threads that are running. If the thread pool is too small, requests might be blocked. If the thread pool is too large, memory might be used unnecessarily, and CPU contention might increase, limiting the processing effectiveness.

We recommend allowing for 10 to 25% more threads than are typically active while the system executes a normal or expected load. Because most of your users will not be active at the same time, set the minimum threads to 10% above the average number of active threads that are observed during monitoring.

We also recommend setting the maximum number of threads to 25% above the average number of active threads that are observed while monitoring under expected load conditions. Make certain to weigh this recommendation against the observed CPU utilization metrics and the suggestions in the PingFederate Performance Tuning Guide.

Logging, reporting, and troubleshooting

This section provides an overview of the available logging, reporting, and troubleshooting features for PingFederate.

PingFederate Logs

The `server.log` file represents the primary troubleshooting log. Along with an HTTP trace from the browser, which can be generated from a debugging application like Fiddler, this file is helpful for identifying issues that must be resolved. The following table identifies the available PingFederate logs and summarizes their purposes.

Name	Purpose
`admin.log`	Records the actions that users of the Administrative Console and the Administrative API perform.
`admin-event-detail.log`	If detailed event logging is enabled, this log records detailed information about each applicable administrative event that users perform using the Administrative Console and the Administrative API.
`admin-api.log`	Records the actions that users of the administrative API perform.
`runtime-api.log`	Records the actions that API users perform by using the OAuth Client Management Service, the OAuth Access Grant Management Service, and the Session Revocation API.
`transaction.log`	Records individual identity-federation runtime transactions at specified levels of detail.
`audit.log`	Records a selected, configurable subset of transaction log information plus additional details. Intended for security-audit and regulatory-compliance purposes.
`provisioner-audit.log`	Records outbound provisioning events intended for security-audit purposes.
`provisioner.log`	Records provisioning activity only. Useful when troubleshooting issues that relate to provisioning.
`server.log`	Records PingFederate runtime and administrative server activities. For more information about the primary troubleshooting log, see Creating an error-only server log.
`init.log`	Records only Jetty messages that are generated prior to starting PingFederate.

Creating an error-only server log

This section describes an approach for modifying your `log4j2.xml` file, which can be sent to a security information and event management (SIEM) tool, such as Splunk. You can configure alerts to send notifications when such events occur, or to improve the monitoring of these events.

About this task

We recommend using the `server.log` file for error-level messages. Even when levels are down to a minimum, the server log generates large amounts of information in an active production environment. As an alternative, you can set up a specific log to log only `ERROR` and higher.

To change your `log4j2.xml` file to enable a separate log file:

Steps

1. Create an appender.

   The easiest way to create an appender is to copy an existing one as a base. In the following example, the `RollingFile` is the same one that the `server.log` file uses. Bold text identifies items that have been changed.

   ```
   [.b]<!-- Error Only Main Log : A size based file rolling appender -->
   <RollingFile name=[.b]"FILEERR" fileName="${sys:pf.log.dir}/server.error.log"
                   filePattern="${sys:pf.log.dir}/server.error.log.%i" ignoreExceptions="false">
           <PatternLayout>
           <!-- Uncomment this if you want to use UTF-8 encoding instead
           of system's default encoding.
           <charset>UTF-8</charset> -->
           <pattern>%d %X{trackingid} %-5p [%c] %m%n</pattern>
           </PatternLayout>
           <Policies>
           <SizeBasedTriggeringPolicy
                   size="10000 KB" />
           </Policies>
           <DefaultRolloverStrategy max="5" />
   </RollingFile>
   ```

2. At the end of your `log4j2.xml` file, set the appender that you created in the previous step for `AsyncRoot`.

   ```
   <AsyncRoot level="INFO" includeLocation="false">
           <!-- <AppenderRef ref="CONSOLE" /> -->
           <AppenderRef ref="FILE" />
            [.b]<AppenderRef ref="FILEERR" level="ERROR" />
   </AsyncRoot>
   ```

   In this example, the `level` attribute indicates the level of messages that are sent to the log file.

3. Remove the attribute `additivity="false"` from all other loggers that contain a reference to the `File` appender.

   ```
   Logger name="org.sourceid.saml20.util.SystemUtil" level="INFO" additivity="false">
           <!--<AppenderRef ref="CONSOLE" /> -->
           <AppenderRef ref="FILE" />
   ```

   Becomes:

   ```
   <Logger name="org.sourceid.saml20.util.SystemUtil" level="INFO" >
           <!--<AppenderRef ref="CONSOLE" /> -->
           <AppenderRef ref="FILE" />
   ```

4. Make this change on all nodes within the cluster.

> **Note**
>
> To expedite this step, we recommend creating a base file with the appropriate changes and copying it to all the nodes.

5. Restart PingFederate.

Splunk dashboards and audit logs

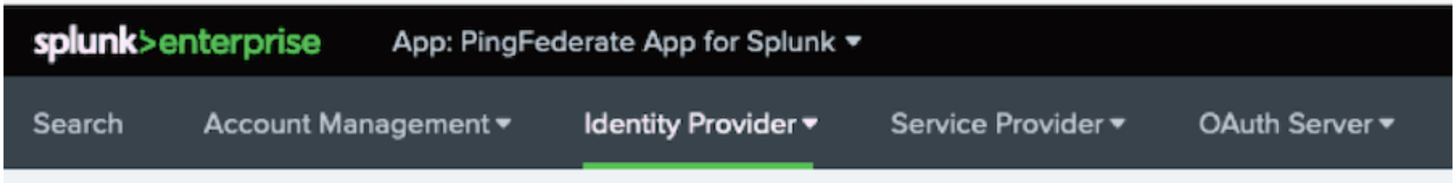
Ping provides a free Splunk for PingFederate application that customers can use to create dashboards.

This application takes advantage of the Writing audit logs for Splunk and Outbound provisioning audit logging, which can be enabled in `log4j2.xml` file.

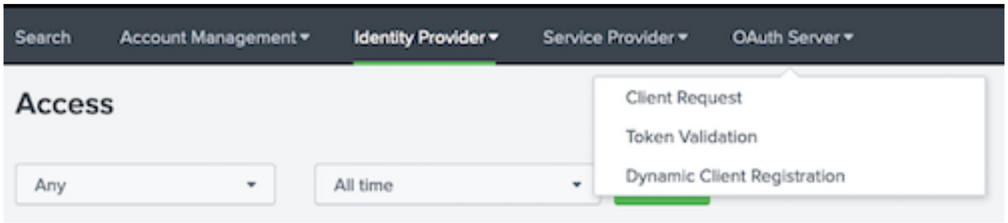
Examples of Splunk dashboards

To help you review different events, the following dashboards are available from the top-level menu of the PingFederate app for Splunk:

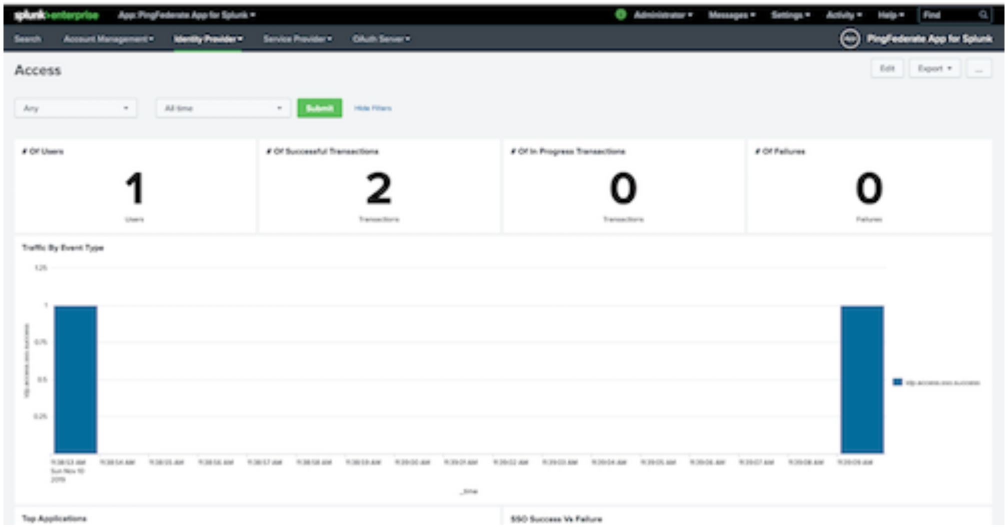
- Account Manager
- Identity Provider
- Service Provider
- OAuth Server



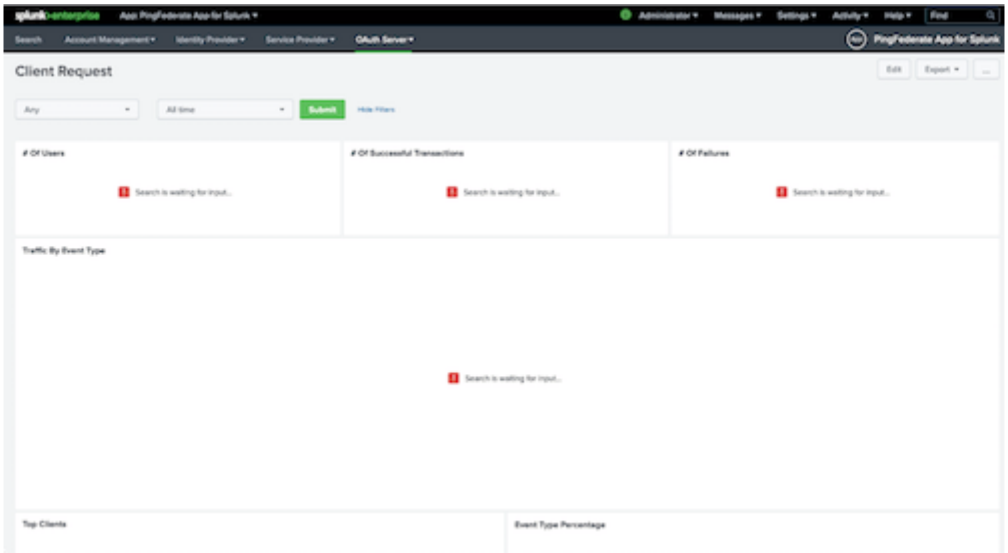
Click a menu item to view its sub-menus, as the following example shows for OAuth Server.



The following image shows the Identity Provider Access sub-menu dashboard with examples from the security audit log entries.



After you select a sub-menu, an image like the following OAuth Server Client Request example is displayed while the dashboard waits for the search results.



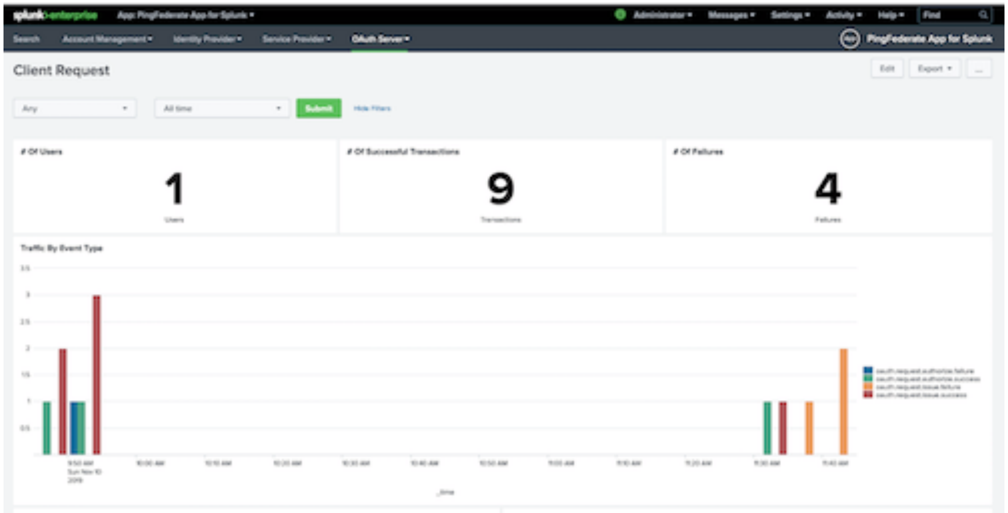
Search is waiting for input...

Search is waiting for input...

Search is waiting for input...

Search is waiting for input...

After you click Submit, the dashboard displays the following search results for the client request.

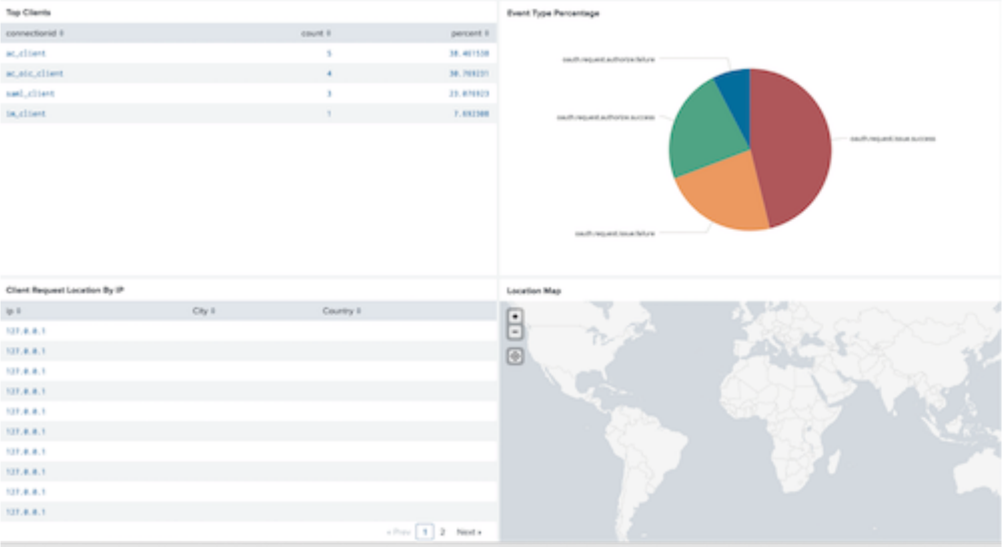


Client Request

Any · All time

# Of Users	# Of Successful Transactions	# Of Failures
1 Users	9 Transactions	4 Failures

To view additional results, scroll downward or select another page. The following Client Request page provides an example.



Top Clients

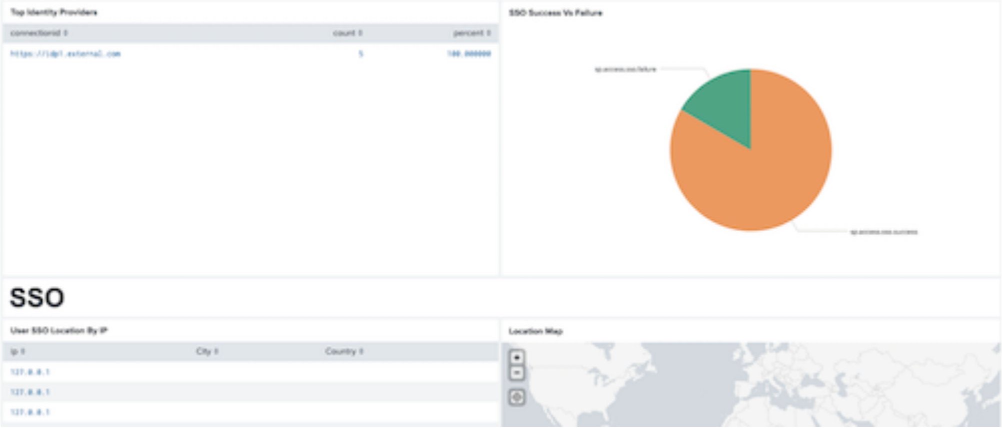
connectionid	count	percent
ac_client	5	38.461538
ac_oic_client	4	30.769231
saml_client	3	23.076923
im_client	1	7.692308

Event Type Percentage
oauth.request.authorize.failure
oauth.request.authorize.success
oauth.request.issue.success
oauth.request.issue.failure
Location Map

Client Request Location By IP

ip	City	Country
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		
127.0.0.1		

The following images provide additional examples of the Service Provider Access sub-menu dashboard.



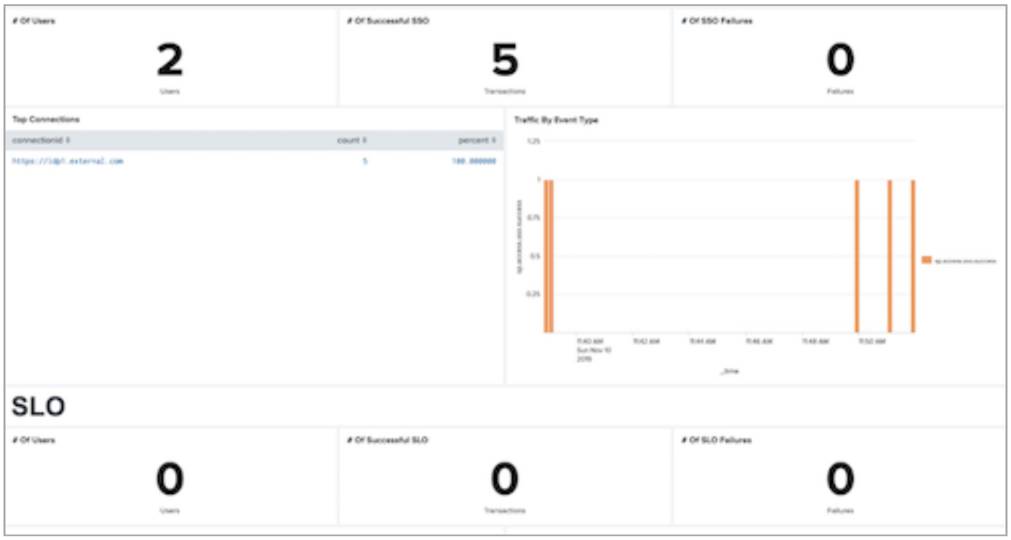
Top Identity Providers

connectionid	count	percent
https://idp1.external.com	5	100.000000

SSO

User SSO Location By IP

ip	City	Country
127.0.0.1		
127.0.0.1		
127.0.0.1		



Top Connections

connectionid ⇕	count ⇕	percent ⇕
https://idp1.external.com	5	100.000000

SDK Developer’s Guide

The PingFederate SDK enables integration with identity providers (IdPs) and service providers (SPs). The SDK allows application developers and system administrators to build custom implementations for communicating authentication and security information between PingFederate and their enterprise environment.

Possible customizations

Extending PingFederate could include:

- Authentication adapters to integrate web applications or identity-management systems
- Authentication selectors to direct single sign-on (SSO) authentication to instances of authentication adapters based on specified conditions
- WS-Trust Security Token Service (STS) token translators, including token processors and token generators
- Custom data source drivers
- Password credential validators
- Identity store provisioners
- Notification publishers

The PingFederate Java SDK consists of several APIs, including:

- Adapter and STS token-translator interfaces
- Authentication selector interfaces
- Custom data source interfaces
- Password credential validator interfaces
- Identity store provisioner interfaces
- Notification publisher interface

These interfaces allow users to create their own custom PingFederate plugins to suit their organization's needs. This SDK provides a means to develop, compile, and deploy custom plugins to PingFederate. The package also contains example plugins for reference. These example plugin projects are located in the `<pf_install>/sdk/plugin-src` directory.

The PingFederate Integration overview describes the pre-built authentication adapters Ping Identity provides for integrating web applications and identity-management systems with PingFederate. Review this document before building your own adapter to see if an available adapter fits your use case.

> **Important**
>
> Custom components might not work the same way after upgrading PingFederate. When upgrading, ensure you thoroughly retest the behavior of customizations in a non-critical upgraded environment.

Adapter and STS token-translator interfaces

The adapter and token-translator APIs enable PingFederate integration with IdPs or SPs. Adapter token-translator APIs are configurable UI plugins that provide requisite runtime integration and allow you to render custom configuration windows.

> **Note**
>
> Suitable adapter or token-translator implementations for your deployment might already exist. Before developing your own custom solution, see the Ping Identity Downloads website for available implementations.

Authentication selector interfaces

Authentication selectors provide a mechanism to choose among multiple authentication sources and to direct a user to use a particular adapter or IdP connections. For example, an authentication selector might map internal corporate users to use one adapter and map external non-corporate users to a different adapter. Authentication selectors are configurable UI plugins that allow you to render custom configuration windows.

Custom data source interfaces

The custom data source API is a set of Java interfaces that enable PingFederate to integrate with datastores not covered by existing Java Database Connectivity (JDBC) or LDAP drivers. This allows developers to retrieve attributes from their choice of data source during attribute fulfillment. Custom data source interfaces are configurable UI plugins that allow you to render custom configuration windows.

Password credential validator interfaces

The password credential validator interfaces allow developers to define credential validators that verify a given username and password in various contexts throughout the system. For example, credential validators are used to configure OAuth Resource Owner authorization grants and the HTML Form Adapter.

Identity store provisioner interfaces

Identity store provisioners provide a mechanism for provisioning and deprovisioning users to external user stores. For example, you can configure a custom identity store provisioner within an inbound provisioning IdP connection to provision users using the System for Cross-domain Identity Management (SCIM) protocol. Identity store provisioners are configurable UI plugins that allow you to render custom configuration windows.

Notification publisher interface

PingFederate delivers messages to administrators and end users based on notification publisher settings. Developers can implement custom notification publishers using the `NotificationPublisherPlugin` interface.

Additional documentation

- Javadocs provide detailed reference information for developers. The Javadocs are located in the `<pf_install>/pingfederate/sdk/doc` directory.
- The user guides for Java, .NET, and PHP integration kits show examples of SDK implementations.

SDK directory structure

This topic describes the directory and build components that comprise the SDK.

The PingFederate SDK directory (`<pf_install>/pingfederate/sdk`) contains the following:

- `plugin-src/` — The directory where you place your custom plugin projects. This directory also contains example plugin implementations showing a wide range of functionality. You can use these examples to develop your own implementations.
- `doc/` — Contains the SDK Javadocs. Open `index.html` to get started.
- `lib/` — Contains libraries used for compiling and deploying custom components into PingFederate.
- `build.properties` — Contains properties used by the Ant build script, `build.xml`, to compile and deploy your custom components. Do not modify this file. If you need to override a property, use `build.local.properties`.
- `build.local.properties` — Allows you to specify which project you want to build and define properties specific to your environment. Use this file to declare the project you want to build.
- `build.xml` — The Ant build script used to compile, build, and deploy your component. This file should not need modification.

The Java SDK, along with Apache Ant, enables you to create directories for your project and use the build script to build, clean, or deploy it. For more information, see Developing your own plugin.

Developing your own plugin

You can set up a development environment within the SDK and use it to create a plugin.

Before you begin

Ensure you have the Java SDK and Apache Ant installed.

> **Important**
>
> The PingFederate SDK only supports JDK 11.

About this task

The Java SDK, along with Apache Ant, enables you to create directories for your project and use the build script to build, clean, or deploy it.

Steps

1. Create a new project directory in the `<pf_install>/pingfederate/sdk/plugin-src` directory.
2. In the new project directory, create a subdirectory named `java`.

   This is where you place the Java source code for your implementation. Follow standard Java package and directory structure layout.
3. If your project depends on third-party libraries, create another subdirectory called `lib` and place the necessary `.jar` files in it.
4. Edit the `build.local.properties` file and set `target-plugin-name` to specify the name of the directory that contains your project.
5. Run the appropriate target to clean, build, or deploy your plugin.

   To display a list of available build targets, run `ant` from `<pf_install>/pingfederate/sdk`.

   ```
   [java] Main targets:
   [java]
   [java] clean-plugin Clean the plug-in build directory
   [java] deploy-plugin Deploy the plug-in jar and libs to PingFederate
   [java] jar-plugin    Package the plug-in jar
   [java]
   [java] Default target: help
   ```

   > **Note**
   >
   > Because it packages the `.jar` files with additional metadata to make them discoverable by PingFederate, we recommend building the project with the `build.xml` file included in the SDK.

Implementation guidelines

The following topics provide programming guidance for developing custom interfaces.

- Shared plugin interfaces
- Developing IdP adapters
- Developing authentication API-capable adapters and selectors
- Developing SP adapters
- Developing token processors
- Developing token generators
- Developing authentication selectors
- Developing data source connectors
- Developing password credential validators

- Developing identity store provisioners
- Developing notification publishers
- Building and deploying with Ant
- Building and deploying manually
- Log messages

For more details about interfaces discussed here and additional functionality, see the SDK Javadocs.

Shared plugin interfaces

Plugin implementations generally invoke methods categorized as either configurable or describable. This document describes these types of plugins and how they are used in PingFederate.

Configurable plugin

Any custom plugin that requires UI settings is configurable and implements the `ConfigurablePlugin` interface. This ensures that PingFederate loads the plugin instance with the correct configuration settings.

All plugin types implement the `ConfigurablePlugin` interface and must define the following within the `ConfigurablePlugin` interface to enable configuration loading.

```
void configure(Configuration configuration)
```

During processing of a configurable plugin instance, PingFederate calls the `ConfigurablePlugin.configure()` method and passes a `Configuration` object. The `Configuration` object provides the plugin adapter instance configuration set by an administrator in the PingFederate UI.

The `SpAuthnAdapterExample.java` sample provided with the SDK shows how to use this method to initialize an adapter instance from a saved configuration. After your implementation loads the configuration values, the plugin instance can use them in other method calls.

Describable plugin

Any plugin that requires configuration windows in the PingFederate administrative console is a describable plugin. Most plugins implement the `DescribablePlugin` interface to ensure that PingFederate renders the correct UI components based on the returned `PluginDescriptor`.

Adapter and custom data source plugins are special cases and do not implement the `DescribablePlugin` interface. However, they still return a plugin descriptor (`AuthnAdapterDescriptor` and `SourceDescriptor`) and are still describable plugins.

All describable plugins must define a UI descriptor. Use one of the following methods to implement a UI descriptor, depending on the type of plugin:

- For plugins using the `DescribablePlugin` interface

  ```
  PluginDescriptor getPluginDescriptor()
  ```

- For adapter plugins

  ```
  AuthnAdapterDescriptor getAdapterDescriptor()
  ```

- For custom data source plugins

  ```
  SourceDescriptor getSourceDescriptor()
  ```

Describable plugins can return a subclass of `PluginDescriptor`, so the return type might differ between plugin implementations. Your plugin implementation populates `PluginDescriptor` with `FieldDescriptors`, `FieldValidators`, and `Actions` and is presented as a set of UI components in the PingFederate administrative console.

> **Tip**
>
> Some plugin types offer concrete descriptor implementations for developers. The Javadocs and examples provided with the SDK show which descriptor classes are available for each plugin type. The examples also show you how to use `FieldDescriptors`, `FieldValidators`, and `Actions` to define your plugin descriptor.

Developing IdP adapters

Developers can use the PingFederate SDK to create identity provider (IdP) adapters.

IdP adapters facilitate integration with external authentication systems, allowing user attributes to be looked up and sessions to be terminated during logout. In addition to performing the primary authentication step, IdP adapters can also perform secondary authentication steps to confirm the user's identity. Administrators can configure IdP adapters in sign-on authentication policies and in policies for resetting or changing passwords.

The IdP authentication adapter interface

Developers can create an IdP adapter by implementing the `IdpAuthenticationAdapterV2` interface. Implementing this interface requires the following Java packages:

- `org.sourceid.saml20.adapter.idp.authn`
- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`

For each IdP adapter implementation, in addition to the methods described in Shared plugin interfaces, you must define methods for:

- User attribute lookup
- Session termination

Looking up user attributes

PingFederate invokes the `lookupAuthN()` method of your IdP adapter to look up user attributes for a request, regardless of whether the request is for IdP- or SP-initiated single sign-on (SSO), an OAuth transaction, or direct IdP-to-SP adapter processing. PingFederate uses the same method to authenticate a user when the adapter is placed in a password reset or password change policy.

```
AuthnAdapterResponse lookupAuthN(
  HttpServletRequest req,
  HttpServletResponse resp,
  Map<String, Object> inParameters)
  throws AuthnAdapterException, IOException;
```

> **Note**
>
> The `IdpAuthenticationAdapterV2` interface provides an overloaded version of the `lookupAuthN()` method. The original `lookupAuthN()` method in the `IdpAuthenticationAdapter` interface is deprecated, so only use the method in the `IdpAuthenticationAdapterV2` interface.

The parameters for the `lookupAuthN()` method are `HttpServletRequest`, the `HttpServletResponse`, and the `inParameters` map. The `inParameters` map contains important information about the SSO transaction that an adapter can use. The parameter keys are defined and documented in the `IdpAuthenticationAdapterV2` interface.

If an adapter is able to identify the user without further interaction, it can immediately return an `AuthnAdapterResponse` containing the user attributes. If the adapter needs further interaction, it can write to the `HttpServletResponse` as appropriate and commit it. Immediately after invoking `lookupAuthN()`, PingFederate checks if the response has been committed. If it has been committed, PingFederate saves the state it needs and stops processing the current transaction. PingFederate continues processing the transaction when the browser returns to the transaction's resume path, at which point PingFederate again invokes the `lookupAuthN()` method. This series of events will be repeated until the method returns without committing the response. When that happens, which could be during the first invocation, PingFederate continues transaction processing using the result of the method.

> **Note**
>
> If the response is committed, then PingFederate ignores the return value. It only uses the return value of an invocation that does not commit the response.

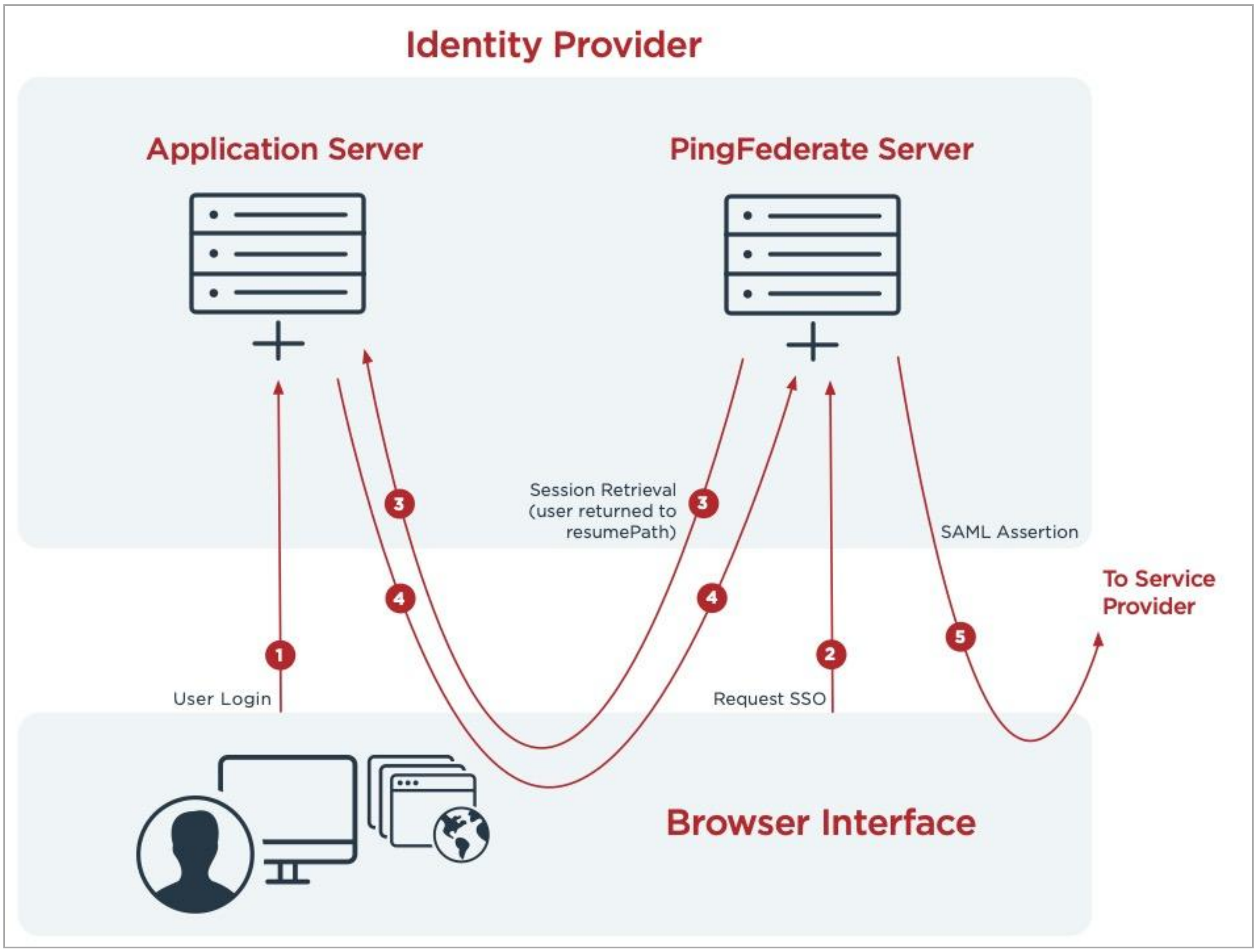
The return type of `lookupAuthN()` is `AuthnAdapterResponse`. When the adapter has completed its processing and returns a value, it must populate the `authnStatus` field of this object. If the `authnStatus` is `SUCCESS`, it must also provide the user attributes in the `attributeMap` field. There are several attribute keys with special meanings, which are defined and documented in the `IdpAuthenticationAdapterV2` interface.

Two basic styles of IdP adapter are common. In the first style, the adapter presents the user a form in which the user enters their credentials to authenticate themselves. Such adapters can use the `TemplateRendererUtil` class to render an HTML template in the user's browser. An example of such an adapter is `template-render-adapter-example`, which is in the `/sdk/plugin-src` directory of PingFederate. That example also shows how you can develop an interactive adapter that supports the PingFederate authentication API. For more information, see Developing authentication API-capable adapters and selectors.

The second style of IdP adapter redirects the user to an external authentication system. After the user has been authenticated, the external system must redirect the user to the resume path for the SSO transaction. This path is provided to the adapter in the `inParameters` map using the key `IN_PARAMETER_NAME_RESUME_PATH`.

How the user attributes are returned to the adapter is up to the implementation. If the authentication system provides an API for retrieving user attributes, a reference to the attributes can be passed through a query parameter appended to the resume URL. Alternatively, if the authentication system shares a parent domain with PingFederate, a cookie can be used to communicate the user attributes.

The following diagram and numbered list show the request sequence for an adapter that redirects to an external authentication system.



IdP-initiated SSO sequence:

1. A user signs on to a local application or domain through an authentication mechanism such as an identity-management system.
2. The user requests access to a protected resource located in the service provider (SP) domain. The link or other mechanism invokes the PingFederate SSO service.
3. PingFederate invokes the designated adapter's lookup method, including the `resumePath` parameter.
4. The application server returns the session information and redirects the browser along with the returned information to the `[.codeph]`resumePath```` URL.

5. PingFederate generates a SAML assertion and sends the browser with the SAML assertion to the SP's SAML gateway.

Managing session attributes

The PingFederate SDK allows adapters to track session attributes. These attributes are stored in memory and replicated across multiple cluster nodes. They are looked up using the attribute name and the `PF` cookie from the user's browser.

Session attributes can be global, spanning multiple SSO transactions, or be linked to a specific transaction. To avoid prompting the user again for credentials, an adapter could use a global session attribute containing the user attributes that were obtained when the user first signed on. A multi-factor authentication (MFA) adapter can use a transaction-scoped attribute to keep track of a one-time passcode that was sent to the user's mobile device. Use `SessionStateSupport` to keep track of global attributes and use `TransactionalStateSupport` to track attributes for the current transaction. It's important to remove transactional attributes when your adapter is finished processing and about to return a result from `lookupAuthN()`. This reduces heap usage and, more importantly, helps avoid subtle security issues.

PingFederate can maintain an authentication session on behalf of your adapter. If an authentication session is enabled for your adapter and user attributes were obtained through a previous call to `lookupAuthN()`, then PingFederate will avoid calling `lookupAuthN()` during subsequent SSO transactions, provided the session idle or maximum timeout has not been exceeded.

Generally, it's better to rely on the PingFederate authentication session capability, rather than implementing your own within your adapter. If you do implement an internal session capability, ensure this session is not used when the authentication policy for the transaction indicates that reauthentication is required. The policy for the transaction is passed in the `inParameters` map using the key `IN_PARAMETER_NAME_AUTHN_POLICY`.

Designing adapters for use in password reset or password change policies

The HTML Form Adapter can invoke an authentication policy during password reset or password change operations. When your adapter is invoked as part of such a policy, `IN_PARAMETER_NAME_USERID` refers to the username the user entered at the start of the operation.

> **Important**
>
> Consider the following when designing adapters that will be used in password reset or password change policies.
> If your adapter can track an internal session, an existing session should not be used during a password reset or password change transaction. PingFederate will ensure the reauthenticate flag in the transaction policy (`IN_PARAMETER_NAME_AUTHN_POLICY` in the `inParameters` map) is set to `true`. When this flag is `true`, your adapter should ignore any existing session.
> Session attributes that track the progress of the current authentication attempt should be stored using `TransactionalStateSupport` and must be removed when returning a success or failure result from `lookupAuthN()`. In particular, if your adapter tracks a session attribute indicating the authentication was successful (for example, to render a "Success" page to the user), you should store that attribute using `TransactionalStateSupport` and ensure it's removed before returning `AUTHN_STATUS.SUCCESS` from `lookupAuthN()`.

Terminating sessions

During single logout (SLO) request processing, PingFederate invokes your IdP adapter's `logoutAuthN()` method to terminate a user's session. This method is invoked during IdP- or SP-initiated SLO requests.

```
boolean logoutAuthN(Map authnIdentifiers,
  HttpServletRequest req,
  HttpServletResponse resp,
  String resumePath)
  throws AuthnAdapterException, IOException
```

Like the `lookupAuthN()` method, the `logoutAuthN()` method has access to the `HttpServletRequest` and `HttpServletResponse` objects. The user attributes returned earlier from `lookupAuthN()` are also passed as the input parameter `authnIdentifiers`.

If your adapter maintains an internal session, it should remove associated session attributes during `logoutAuthN()`. If the adapter is associated with an external authentication system, it can redirect the browser to an endpoint used to terminate the session at the application. The `resumePath` parameter contains the URL to which the user is redirected to complete the SLO process.

Developing authentication API-capable adapters and selectors

The PingFederate authentication API lets applications interact with authentication policies. Making an adapter or selector plugin API-capable means ensuring that an authentication application can invoke the plugin through this API.

API-capable plugins must handle JSON-based API requests. When a plugin is invoked through the authentication API endpoint, if it needs interaction from the user, the plugin sends a JSON-based response rather than rendering a template.

Adapter and selector plugins handle distinct kinds of requests and generate distinct kinds of responses. The main method you implement in adapters is `lookupAuthN()`. The main method you implement in selectors is `selectContext()`.

Developing an API-capable plugin requires a dependency on the PingFederate authentication API SDK JAR file, `pf-authn-api-sdk-version.jar`. In the PingFederate installation package, you can find the SDK JAR file in the `server/default/lib` directory. Documentation for the classes are in the Javadocs for the standard PingFederate SDK, under `sdk/doc/index.html` in the PingFederate install package.

Related links

- Authentication API states, actions, and models
- Specification of the plugin API
- State model contents
- Non-interactive plugins
- Runtime behavior implementation
- Session state management
- Error messages and localization

Authentication API states, actions, and models

To develop authentication API-capable adapters and selectors, you must understand the states, actions, and models of single sign-on (SSO) transactions through the PingFederate authentication API.

PingFederate assigns a flow ID to each SSO transaction that uses the authentication API. PingFederate uses the flow ID to determine a transaction's state.

As a user steps through an SSO transaction, the transaction is always in some state. The state includes a status field and other fields specific to that state. The class containing those other fields is the model for the state.

The API endpoint returns the following when the user's SSO transaction has reached the `USERNAME_PASSWORD_REQUIRED` state for the form adapter.

```
{
        "id": "PyH5g",
        "pluginTypeId": "7RmQNDWaOnBoudTufx2sEw",
        "status": "USERNAME_PASSWORD_REQUIRED",
        "showRememberMyUsername": false,
        "showThisIsMyDevice": false,
        "thisIsMyDeviceSelected": false,
        "showCaptcha": false,
        "rememberMyUsernameSelected": false,
        "_links": {
                "self": {
                        "href": "https://localhost:9031/pf-ws/authn/flows/PyH5g"
                },
                "checkUsernamePassword": {
                        "href": "https://localhost:9031/pf-ws/authn/flows/PyH5g"
                }
        }
}
```

The model for this state is the class `UsernamePasswordRequired`. It includes fields such as `showThisIsMyDevice`, which help the API client know how to render the credential prompt to the user.

The API response also includes a list of available actions. In this case, the only action available is `checkUsernamePassword`. The API client can select this action by sending a `POST` request with the `Content-Type` of `application/vnd.pingidentity.checkUsernamePassword+json`. Each action has its own model containing the fields that the `POST` body can provide. For the `checkUsernamePassword` action, the model is `CheckUsernamePassword`.

The `POST` body can be as simple as the following.

```
{
        "username": "joe",
        "password": "2Federate"
}
```

After receiving this request, PingFederate calls the `lookupAuthN()` method of the form adapter. If the form adapter encounters an error while validating the credentials, it writes a JSON API error to the response. If the form adapter successfully validates the credentials, it returns `AUTHN_STATUS.SUCCESS` from its `lookupAuthN()` method. PingFederate then goes to the next step in the authentication policy. If the next step is an API-capable adapter, PingFederate calls `lookupAuthN()` on that adapter and the adapter determines its current state and writes it to the response, along with the available actions.

> **Note**
>
> The PingFederate authentication API follows a different naming convention for actions than PingOne. PingOne names actions as `noun.verb`, such as `otp.check`. PingFederate uses `verbNoun`, such as `checkOtp`.

Related links

- Developing authentication API-capable adapters and selectors

Specification of the plugin API

The first step in adding API support to your plugin is to implement the `AuthnApiPlugin` interface.

The AuthnApiPlugin interface has two methods: `getApiSpec()` and `getApiPluginDescriptor()`. You only need to implement the `getApiSpec()` method. The API specification this method returns defines the states, models, and actions that your plugin exposes in the API.

The API specification is defined by the `*Spec` classes in the SDK. These include `AuthnStateSpec`, `AuthnActionSpec`, `AuthnErrorSpec`, and `AuthnErrorDetailSpec`. The information in these classes lets the PingFederate authentication API Explorer provide documentation for API client developers. That documentation describes your plugin's API and lets developers experiment with it.

You can access the API Explorer at https://*PingFederate_host*:9031/pf-ws/authn/explorer. To enable the API Explorer, go to the Authentication API Applications window and select the Enable API Explorer check box. An easy way to use the API Explorer is to create an authentication API application in PingFederate and set the URL for the application to the API Explorer's URL.

> **Note**
>
> When defining models for states and actions, use the @Schema annotation to describe each field in the model and show whether the field is required.

The rest of this document primarily uses the `TemplateRenderAdapter` as an example. The source for this adapter is in the PingFederate installation package's `sdk/plugin-src/template-render-adapter-example` directory. This adapter is simple. It just prompts the user to enter their username and provide a set of string attributes. The administrator defines the list of attributes by extending the adapter contract. The attribute values are passed back in the `SubmitUserAttributes` model as a map. Representing field values using a map in the model is unusual. Usually a separate field in the model defines each allowed field, which provides better type safety in the code.

State model example

The following is an example of a state model used by the `TemplateRenderAdapter`.

```
/**
 * This is the model for the USER_ATTRIBUTES_REQUIRED state. It defines the
 * fields that are returned to the API client in a GET response for this state.
 */
public class UserAttributesRequired
{
        private List<String> attributeNames = new ArrayList<>();

        /**
         * Get the list of user attributes supported by this adapter instance.
         *
         * It is recommended to annotate each getter with the @Schema annotation
         * and provide a description. This description will be used in
         * generating API documentation.
         */
        @Schema(description="A list of user attribute names that are supported by this adapter instance.")
        public List<String> getAttributeNames()
        {
                return attributeNames;
        }

        /**
         * Set the list of user attributes supported by this adapter instance.
         */
        public void setAttributeNames(List<String> attributeNames)
        {
                this.attributeNames = attributeNames;
        }
}
```

Action model example

The following is the model for the `submitUserAttributes` action.

```
/**
 * This is the model for the submitUserAttributes API action. It defines the
 * fields that may be included in the POST body for this action.
 */
public class SubmitUserAttributes
{
        private String username;
        private Map<String,Object> userAttributes = new HashMap<>();

        /**
         * Get the username.
         *
         * It is recommended to annotate each getter with the @Schema annotation
         * and provide a description. The 'required' flag can also be specified. This
         * information will be used in generating API documentation.
         */
        @Schema(description="The user's username.", required=true)
        public String getUsername()
        {
                return username;
        }

        /**
         * Set the username.
         */
        public void setUsername(String username)
        {
                this.username = username;
        }

        /**
         * Get the user attributes.
         */
        @Schema(description="Additional user attributes, as name-value pairs.")
        public Map<String, Object> getUserAttributes()
        {
                return userAttributes;
        }

        /**
         * Set the user attributes.
         */
        public void setUserAttributes(Map<String, Object> userAttributes)
        {
                this.userAttributes = userAttributes;
        }
}
```

AuthnStateSpec and AuthnActionSpec objects

A fluent builder is provided for creating `AuthnStateSpec` and `AuthnActionSpec` objects.

Here is the definition of the `AuthnStateSpec` for the `USER_ATTRIBUTES_REQUIRED` state:

```
public final static AuthnStateSpec<UserAttributesRequired> USER_ATTRIBUTES_REQUIRED = new
AuthnStateSpec.Builder<UserAttributesRequired>()
        .status("USER_ATTRIBUTES_REQUIRED")
        .description("The user's username and attributes are required.")
        .modelClass(UserAttributesRequired.class)
        .action(ActionSpec.SUBMIT_USER_ATTRIBUTES)
        .action(CommonActionSpec.CANCEL_AUTHENTICATION)
        .build();
```

Here is the specification for the `submitUserAttributes` action:

```
public final static AuthnActionSpec<SubmitUserAttributes> SUBMIT_USER_ATTRIBUTES = new
AuthnActionSpec.Builder<SubmitUserAttributes>()
        .id("submitUserAttributes")
        .description("Submit the user's username and attributes.")
        .modelClass(SubmitUserAttributes.class)
        .error(CommonErrorSpec.VALIDATION_ERROR)
        .errorDetail(ErrorDetailSpec.INVALID_ATTRIBUTE_NAME)
        .build();
```

Error specifications

Action specifications can include a list of possible errors and error details. Each top-level error that an authentication API request returns can include one or more error detail objects underneath it.

Typically, in the API specification, your only top-level error will be `CommonErrorSpec.VALIDATION_ERROR`. However, you can include error detail specifications that can appear under that top-level error.

The following is how the `TemplateRenderAdapter` defines the specification for the `INVALID_ATTRIBUTE_NAME` error detail.

```
public final static AuthnErrorDetailSpec INVALID_ATTRIBUTE_NAME = new AuthnErrorDetailSpec.Builder()
        .code("INVALID_ATTRIBUTE_NAME")
        .message("An invalid attribute name was provided.")
        .parentCode(CommonErrorSpec.VALIDATION_ERROR.getCode())
        .build();
```

> **Note**
>
> The error detail specification must reference the error code of its parent top-level error. This ensures that the authentication API Explorer correctly represents the error information.

`INVALID_ATTRIBUTE_NAME` is an example of an error that would be useful for API client developers but not for end users.

For more information about defining user-facing errors, see Error messages and localization.

Related links

- Developing authentication API-capable adapters and selectors

State model contents

The model for a state includes all the information an API client would need to build a form (not necessarily an HTML form) to show the user. A state model should not include the text for messages to display to the user.

Defining messages for users, and localizing them if needed, is the responsibility of the API client. One of the reasons we avoid including messages in state models is that those messages will often end up including semantic content that the API client needs to drive its code. Following the rule that models do not include messages helps ensure that our models include all the fields that an API client needs to provide the desired user experience.

Sometimes following this rule requires you to add more states. This is preferable to embedding the state information inside of a message because it makes it easier for an API client to handle that state in the desired way.

The one exception we have to this rule is around error messages. API errors include error message text, and in some cases, API clients will display the message text directly to users. This avoids every API client having to write its own messages for every user-facing error the API can generate. For more information, see Error messages and localization.

Related links

- Developing authentication API-capable adapters and selectors

Non-interactive plugins

Some plugins, typically selectors, do not need to interact with the user to do their job. Making these plugins API-capable is straight-forward.

You still implement the `AuthnApiPlugin` interface, but you can just return null from the `getApiSpec()` method. And then you override the default implementation of `getApiPluginDescriptor()` and return an `AuthnApiPluginDescriptor` instance with the `interactive` flag set to `false`. As with many other classes in the SDK, there is an `AuthnApiPluginDescriptor.Builder` class to help in creating the descriptor.

When `interactive` is `false`, PingFederate knows that it never needs to redirect when it encounters your selector. If the request is occurring on the API endpoint, PingFederate can immediately call `selectContext()`. The same is true if the request is occurring on a front-channel endpoint, such as `/as/authorization.oauth2`.

If your selector does not implement `AuthnApiPlugin`, then PingFederate assumes that only a front-channel endpoint can call your selector. If PingFederate encounters your selector while executing an API request, PingFederate will send a `RESUME` response to the API client so that the user is redirected to PingFederate.

Related links

- Developing authentication API-capable adapters and selectors

Runtime behavior implementation

After you specify your plugin's API at least partially, you can start implementing the runtime behavior. Use the specification that you defined previously to implement the runtime functionality.

Follow this pattern in `lookupAuthN()`:

1. Check for the possible actions the adapter expects in the current state.

2. If an action is matched, then try to extract the expected model from the request and handle the action.
3. If an action is requested, but it does not match an action allowed for the current state, then return an `INVALID_ACTION_ID` error.
4. If no action is requested, render the response for the current state.

The `AuthnApiSupport` class provides much of the functionality for handling API requests and sending responses. The `TemplateRenderAdapter` stores a reference to this singleton in its `apiSupport` field.

```
private AuthnApiSupport apiSupport = AuthnApiSupport.getDefault();
```

Checking for actions

The following code shows the preferred approach for checking for the `submitIdentifiers` action.

The adapter performs this check two ways, depending on whether the current request is from the API endpoint. The `TemplateRenderAdapter` uses a slightly different but equivalent method.

```
/**
 * Determine if the user chose "Submit".
 */
private boolean isSubmitAttributesRequest(HttpServletRequest req)
{
        if (apiSupport.isApiRequest(req))
        {
                return ActionSpec.SUBMIT_USER_ATTRIBUTES.isRequested(req);
        }
        return StringUtils.isNotBlank(req.getParameter("pf.submit"));
}
```

Extracting models from requests

The next step extracts the model from the request. This step varies depending on whether the request is from the API endpoint. For an API request, call the `AuthnApiSupport.deserializeAsModel()` method. For a non-API request, you must build the model from the parameters in the request.

```
private SubmitUserAttributes getSubmittedAttributes(HttpServletRequest req) throws AuthnErrorException,
AuthnAdapterException
{
        if (apiSupport.isApiRequest(req))
        {
                try
                {
                        return apiSupport.deserializeAsModel(req, SubmitUserAttributes.class);
                }
                catch (IOException e)
                {
                        throw new AuthnAdapterException(e);
                }
        }
        else
        {
                SubmitUserAttributes result = new SubmitUserAttributes();
                result.setUsername(req.getParameter("username"));

                for (String key : extendedAttr)
                {
                        result.getUserAttributes().put(key, req.getParameter(key));
                }
                return result;
        }
}
```

The `deserializeAsModel()` method also does some validation on the incoming JSON. This includes checking for fields flagged as `required` in the model, using the @Schema annotation. If a validation error occurs during this step, the method throws an `AuthnErrorException`, which the adapter can convert to an API error response. For more information, see Handling authentication error exceptions.

Performing additional validation

The `deserializeAsModel()` method performs some basic validation on the submitted JSON. Your adapter probably needs to perform more validation and send an `AuthnError` to the API client if it finds any errors. Here is how the `TemplateRenderAdapter` validates the names of the provided user attributes:

```
private void validateSubmittedAttributes(HttpServletRequest req, SubmitUserAttributes submitted) throws
AuthnErrorException
{
        if (apiSupport.isApiRequest(req))
        {
                List<AuthnErrorDetail> errorDetails = new ArrayList<>();
                for (String attrName : submitted.getUserAttributes().keySet())
                {
                        if (!extendedAttr.contains(attrName))
                        {

errorDetails.add(ErrorDetailSpec.INVALID_ATTRIBUTE_NAME.makeInstanceBuilder()
                                        .message("Invalid attribute name: " + attrName).build());
                        }
                }
                if (!errorDetails.isEmpty())
                {
                        AuthnError authnError = CommonErrorSpec.VALIDATION_ERROR.makeInstance();
                        authnError.setDetails(errorDetails);
                        throw new AuthnErrorException(authnError);
                }
        }
}
```

Handling invalid action IDs

If a request from an API client includes an action ID that does not match any actions available in the current state, it is best practice to return an error to the client.

After checking all the possible actions, if none match and the request's action ID is not null, the adapter can throw an `AuthnErrorException`. The adapter catches this exception and writes an error to the API response.

```
if (apiSupport.getActionId(req) != null)
{
        // An action ID was provided but it does not match one of those expected in the current state.
        throw new AuthnErrorException(CommonErrorSpec.INVALID_ACTION_ID.makeInstance());
}
```

Handling authentication error exceptions

If the `deserializeAsModel()` method detects an error while deserializing the model, it throws an `AuthnErrorException`. If the added validation checks in `validateSubmittedAttributes` detect an error, they also throw this exception.

The adapter should catch this exception and send an API error response using the method `AuthnApiSupport.writeErrorResponse()`.

```
try
{
        ...
}
catch (AuthnErrorException e)
{
        // A validation error occurred while processing an API request, return an error response to the API
client
        apiSupport.writeErrorResponse(req, resp, e.getValidationError());
        authnAdapterResponse.setAuthnStatus(AUTHN_STATUS.IN_PROGRESS);
        return authnAdapterResponse;
}
```

Sending API responses

`AuthnApiSupport` provides several methods for writing API responses.

The following example shows how the `TemplateRenderAdapter` writes the response for the `USER_ATTRIBUTES_REQUIRED` state.

```
private void renderApiResponse(HttpServletRequest req, HttpServletResponse resp, Map<String, Object>
inParameters) throws AuthnAdapterException
{
        UserAttributesRequired model = new UserAttributesRequired();
        model.setAttributeNames(new ArrayList<>(extendedAttr));
        AuthnState<UserAttributesRequired> authnState = apiSupport.makeAuthnState(req,
StateSpec.USER_ATTRIBUTES_REQUIRED, model);
        try
        {
                apiSupport.writeAuthnStateResponse(req, resp, authnState);
        }
        catch (IOException e)
        {
                throw new AuthnAdapterException(e);
        }
}
```

The `makeAuthnState()` method takes an `AuthnStateSpec` and an instance of the model for that state and builds an `AuthnState` object. The `AuthnState` object can then be further modified. For example, you could remove an action that is not currently applicable using the `removeAction()` method. Then you write the `AuthnState` object to the response using the `writeAuthnStateResponse()` method.

Returning authentication statuses

As with non-API requests, when the adapter finishes, it returns `AUTHN_STATUS.SUCCESS` or `AUTHN_STATUS.FAILURE` from `lookupAuthN()`.

If the adapter has not yet finished and has written something to the response, it should return `AUTHN_STATUS.IN_PROGRESS`.

Related links

- Developing authentication API-capable adapters and selectors

Session state management

Session state management for authentication API adapters is no different than for regular adapters. The same mechanisms, such as `SessionStateSupport` and `TransactionalStateSupport`, are used to store and retrieve session attributes on behalf of a user.

It should not be necessary to store more state attributes just to support the authentication API. The same session attributes should cover both API and non-API requests.

You can also wrap an API-capable adapter in a PingFederate-managed authentication session. PingFederate-managed authentication sessions mean that many adapters no longer need to provide their own internal session tracking.

Related links

- Developing authentication API-capable adapters and selectors

Error messages and localization

Error messages are the one case where an API response could include user-facing text. The typical case is a validation error.

For validation errors, the adapter constructs an `AuthnError` with the code `VALIDATION_ERROR`, and then adds `AuthnErrorDetail` objects for each of the errors that occurred. The `userMessage` field of the `AuthnErrorDetail` object provides the user-facing text. Like states and actions, you can define errors up front using an `AuthnErrorSpec` or an `AuthnErrorDetailSpec`. Then an instance of the error is constructed from the specification on demand.

The following example shows how you can define the specification for an invalid OTP error.

```
public final static AuthnErrorDetailSpec INVALID_OTP = new AuthnErrorDetailSpec.Builder()
         .code("INVALID_OTP")
         .message("An invalid or expired OTP code was provided.")
         .userMessage("This code is invalid or has expired.")
         .parentCode(CommonErrorSpec.VALIDATION_ERROR.getCode())
         .build();
```

The following example shows how you can use that specification to send an error response to the API client.

```
AuthnErrorDetail errorDetail = ErrorDetailSpec.INVALID_OTP.makeInstanceBuilder().build();
AuthnError authnError = CommonErrorSpec.VALIDATION_ERROR.makeInstanceBuilder().detail(errorDetail).build();
apiSupport.writeErrorResponse(req, resp, authnError);
```

To localize the error message using a properties file for your adapter, you can use `LocaleUtil` and `LanguagePackMessages` from the standard PingFederate SDK.

```
LanguagePackMessages messages = new LanguagePackMessages("my-adapter-messages",
LocaleUtil.getUserLocale(req));
String errorMessage = messages.getMessage("invalid.otp.key", new String[]{});
AuthnErrorDetail errorDetail =
ErrorDetailSpec.INVALID_OTP.makeInstanceBuilder().userMessage(errorMessage).build();
AuthnError authnError = CommonErrorSpec.VALIDATION_ERROR.makeInstanceBuilder().detail(errorDetail).build();
apiSupport.writeErrorResponse(req, resp, authnError);
```

For more information about how PingFederate determines the user's locale at runtime, see Locale overrides by cookies.

Some errors reflect problems with API client programming rather than with end user input. If you think an error will not be shown to an end user, then you do not need to populate the `userMessage` field.

Related links

- Developing authentication API-capable adapters and selectors

Developing SP adapters

This topic describes how to create a service provider (SP) adapter, as well as the methods used during SP session creation, SP adapter session logout, and SP account linking.

SP authentication adapter interface

Create service provider (SP) adapters by implementing the `SPAuthenticationAdapter` interface. Implementing this interface requires the following Java packages:

- `org.sourceid.saml20.adapter.sp.authn`
- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`

For each SP adapter implementation, in addition to the methods described in Shared plugin interfaces, you must define:

- SP session creation
- SP adapter session logout
- SP account linking

SP session creation

PingFederate invokes the `createAuthN()` method during the processing of a single sign-on (SSO) request to establish a security context in the external application for the user.

```
java.io.Serializable createAuthN(SsoContext ssoContext,
  javax.servlet.http.HttpServletRequest req,
  javax.servlet.http.HttpServletResponse resp,
  java.lang.String resumePath)
```

This method resembles the `IdpAuthenticationAdapter.lookupAuthN()` method in terms of the objects passed to it and its support for asynchronous requests using the `HttpServletResponse` and `resumePath` parameters. It also accepts an `SsoContext` object, which has access to information such as user attributes and the target destination URL.

SP adapter session logout

PingFederate invokes the `logoutAuthN()` method during a single logout (SLO) request to terminate a user's session with the external application.

```
boolean logoutAuthN (java.io.Serializable authnBean,
  javax.servlet.http.HttpServletRequest req,
  javax.servlet.http.HttpServletResponse resp,
  java.lang.String resumePath)
  throws AuthnAdapterException, java.io.IOException
```

The `HttpServletResponse` and `resumePath` objects are available to support scenarios where the user's browser redirects to an additional service to clean up any remaining sessions.

SP account linking

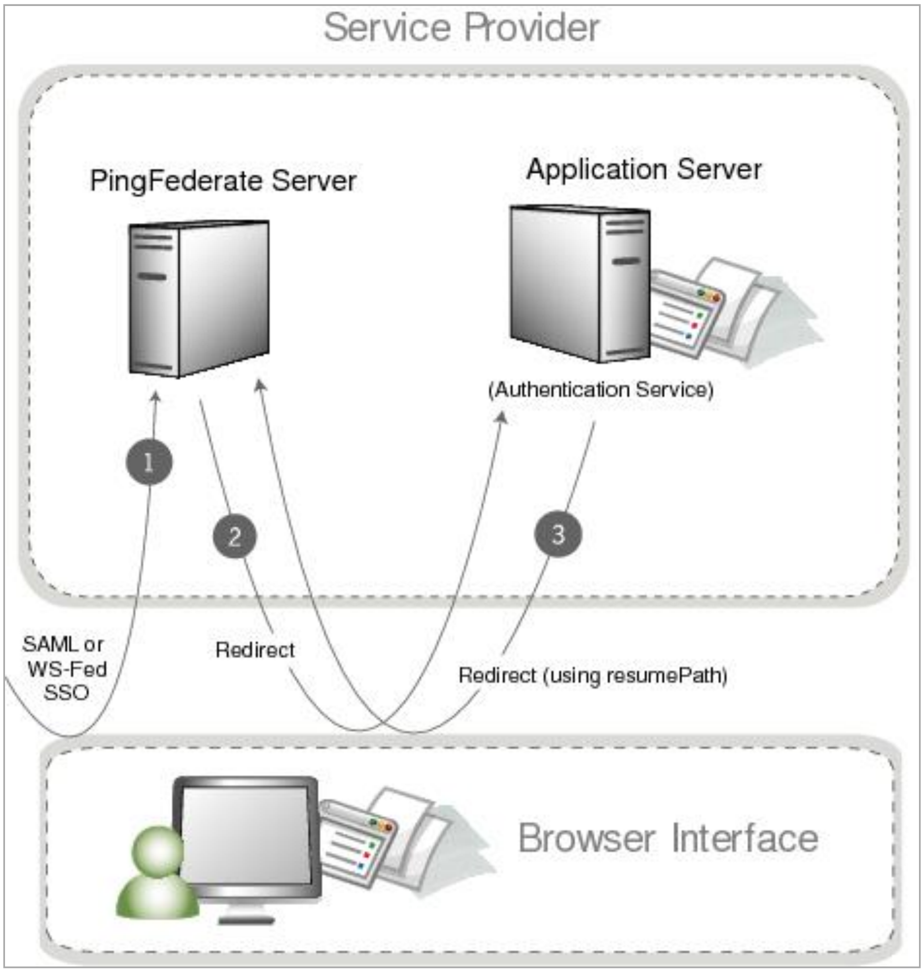
PingFederate invokes the `lookupLocalUserId()` method during an SSO request when the identity provider (IdP) connection uses account linking but no account link for this user is yet established.

```
java.lang.String lookupLocalUserId(
    javax.servlet.http.HttpServletRequest req,
    javax.servlet.http.HttpServletResponse resp,
    java.lang.String partnerIdpEntityId,
    java.lang.String resumePath)
    throws AuthnAdapterException, java.io.IOException
```

After the account link is set, PingFederate maintains this information until the user defederates, which occurs when the user clicks a hyperlink redirecting them to the `/sp/defederate.ping` PingFederate endpoint.

The `HttpServletResponse` and `resumePath` objects are used to send the user to a local service where the user authenticates. After authentication, the user is redirected to the URL specified in the `resumePath` parameter and PingFederate completes the account link.

The following diagram illustrates a typical account-link sequence.



Use the `HttpServletRequest` to read a local session token. The `lookupLocalUserId()` method should return a local user identifier `String` object.

Developing token processors

You can create a token processor by implementing the `TokenProcessor` interface.

The following Java packages are required for implementing the `TokenProcessor` interface:

- `org.sourceid.saml20.adapter.attribute`
- `org.sourceid.saml20.adapter.idp.authn`
- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`
- `org.sourceid.wstrust.model`
- `org.sourceid.wstrust.plugin`
- `org.sourceid.wstrust.plugin.process`
- `com.pingidentity.sdk`

For each token-processor implementation, in addition to the methods described under Shared plugin interfaces, you must define the `TokenContext processToken(T token)` method.

PingFederate invokes the `processToken()` method when processing a security token service (STS) request to perform necessary operations for determining the validity of a token. The type parameter `T` must extend, at a minimum, the type `SecurityToken`. The type `BinarySecurityToken` is also available to represent custom security tokens that can be transported as Base64-encoded data.

Developing token generators

You can create a token-generator implementation by using the `TokenGenerator` interface.

The following Java packages are required for implementing the `TokenGenerator` interface:

- `org.sourceid.saml20.adapter.sp.authn`
- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`
- `org.sourceid.wstrust.model`
- `org.sourceid.wstrust.plugin`
- `org.sourceid.wstrust.plugin.process`
- `com.pingidentity.sdk`

For each token-generator implementation, in addition to the methods described under Shared plugin interfaces, you must define the `SecurityToken generateToken(TokenContext attributeContext)` method.

PingFederate invokes the `generateToken()` method when processing a security token service (STS) request to perform necessary operations for generation of a security token. The type `BinarySecurityToken` is available and you can use it to represent custom security tokens that can be transported as Base64-encoded data. The `TokenContext` contains subject data available for insertion into the generated security token.

Developing authentication selectors

This topic describes aspects of authentication selectors within the context of PingFederate, including implementation, context selection, and callbacks.

Authentication selector interface

Authentication selectors allow PingFederate to choose an appropriate authentication source, an identity provider (IdP) adapter or an IdP connection (for federation hub use cases), based on criteria defined in the authentication selector instance.

When creating an authentication selector, use the following primary Java packages:

- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`

- `com.pingidentity.sdk`

For each authentication selector implementation, in addition to the methods described under Shared plugin interfaces, you must define the following:

- Context Selection
- Authentication selector callback

Context selection

PingFederate calls the `selectContext()` method to determine which authentication source to select. The `mappedAuthnSourcesNames` contains the list of `AuthenticationSourceKeys` and names that are available for the selector to reference.

```
AuthenticationSelectorContext selectContext(HttpServletRequest req,
 HttpServletResponse resp,
 Map<AuthenticationSourceKey, String> mappedAuthnSourcesNames,
 Map<String, Object> extraParameters,
 String resumePath)
```

The `HttpServletRequest` can evaluate cookies, parameters, headers, and other request information to determine which authentication source to select. The `HttpServletResponse` also helps determine the appropriate authentication source to select if the authentication selector requires user interaction. If the `resp` object is written to, it is considered a committed response and returned to the user's browser. The `resumePath` is a relative URL used in conjunction with the `resp` object, such that the user's browser is sent to this URL to resume the single sign-on (SSO) work flow.

After an authentication source is selected, you can create an `AuthenticationSelectorContext` to denote which authentication source to use. You can reference the selected authentication source by its ID or by its context, which is a name that decouples authentication selectors from the configured IDs.

Authentication selector callback

PingFederate calls the `callback()` method after authenticating against a selected source. The `callback()` method allows authentication selectors to update resulting attributes, set cookies, or perform other custom functions.

```
void callback(HttpServletRequest req,
HttpServletResponse resp,
Map authnIdentifiers,
AuthenticationSourceKey authenticationSourceKey,
AuthenticationSelectorContext authnSelectorContext);
```

> **Note**
>
> Writing content to the `resp` object in the `callback()` method is not supported, and doing so might result in unexpected behavior. Setting cookies is supported.

Developing data source connectors

Use PingFederate to query various data sources or build data source connectors to process customized data sources.

PingFederate can query data sources for a variety of purposes using LDAP or Java Database Connectivity (JDBC) interfaces. Use the PingFederate SDK to build data source connectors to query additional data source types. Examples of other data sources include a web service, a flat file, or a different way of using a JDBC or LDAP connection than what is supplied by PingFederate.

The following are the primary Java packages used to build a custom data source:

- `com.pingidentity.sources`
- `com.pingidentity.sources.gui`

For each implementation described in Shared plugin interfaces, you must define the following:

- Connection testing
- Available fields retrieval
- Data source query handling

Data source connection testing

```
boolean testConnection()
```

When associating a custom data source with an identity provider (IdP) or service provider (SP) connection, PingFederate tests connectivity to the data source by calling the `testConnection()` method. Your implementation of this method should perform the necessary steps to demonstrate a successful connection and return `true`, or return `false` if your implementation cannot communicate with the datastore. A `false` result prevents an administrator from continuing with the data source configuration.

Data source available fields retrieval

```
java.util.List<java.lang.String> getAvailableFields()
```

PingFederate calls the `getAvailableFields()` method to determine the available fields that can be returned from a query of this data source. These fields are displayed to the PingFederate administrator during the configuration of a data source lookup, and the administrator selects the attributes from the data source and maps them to the adapter or attribute contract. PingFederate requires at least one field returned from this method.

Data source query handling

```
java.util.Map<java.lang.String,java.lang.Object> retrieveValues(
  java.util.Collection<java.lang.String> attributeNamesToFill,
  SimpleFieldList filterConfiguration)
```

When processing a connection using a custom data source, PingFederate calls the `retrieveValues()` method to perform the actual query for user attributes. This method receives a list of attribute names populated with data. The method can also receive a `filterConfiguration` object populated with a list of fields. Each field contains a name/value pair determined at runtime and collectively used as the criteria for selecting a specific record. In most cases, the criteria are used to locate additional user attributes.

Create the filter criteria selections needed for this lookup by passing back a `CustomDataSourceDriverDescriptor`, an implementation of `SourceDescriptor`, from the `getSourceDescriptor()` method. A `CustomDataSourceDriverDescriptor` can include a `FilterFieldDataDescriptor` composed of a list of fields that can be used as the query criteria. This list of fields is displayed similarly to the other UI-descriptor display fields.

> **Note**
>
> The `filterConfiguration` object is set and populated with a list of fields only if the data source was defined with a `CustomDataSourceDriverDescriptor`. If the `CustomDataSourceDriverDescriptor` was not used in the definition of the data source, the `filterConfiguration` object is set to null.

> **Important**
>
> To pass runtime attribute values to the filter, an administrator must reference the attributes using the `$\{attribute name}` format when defining a filter in the PingFederate administrative console.

After all relevant attributes are retrieved from the data source, they must be returned as a map of name/value pairs, where the names correspond to the initial collection of attribute names passed into the method and the values are the attributes.

Developing password credential validators

Password credential validators allow PingFederate administrators to define a centralized location for username/password validation, allowing PingFederate configurations to reference validator instances.

To implement a custom password credential validator, import the following Java packages:

- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`
- `org.sourceid.util.log`
- `com.pingidentity.sdk`
- `com.pingidentity.sdk.password`

For each implementation, in addition to the methods described under Shared plugin interfaces, you must define the following method.

```
AttributeMap processPasswordCredential(String username,
  String password)
  throws PasswordValidationException
```

This method takes a username and password and verifies the credential against an external source. If the credentials are valid, it returns an `AttributeMap` containing at least one entry representing the principal. If the credentials are invalid, then it returns `null` or an empty map. If the plugin was unable to validate the credentials (for example, due to an offline host or network problems), it returns a `PasswordValidationException`.

To enable password changes in a password credential validator, implement the `com.pingidentity.sdk.password.ChangeablePasswordCredential` interface.

To enable password resets in a password credential validator, implement the `com.pingidentity.sdk.password.ResettablePasswordCredential` interface.

> **Note**
>
> Depending on your password management system, you might need additional system configuration to enable password changes. For example, you can change passwords in Active Directory only if LDAPS is enabled.

Developing identity store provisioners

You can create an identity store provisioner by implementing either the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interface.

Both interfaces support provisioning and deprovisioning users and groups to an external user store. The `IdentityStoreProvisionerWithFiltering` interface supports list/query and filtering; the `IdentityStoreProvisioner` interface does not. For more information about list/query and filtering, see 3.2.2. List/Query Resources and 3.2.2.1. Filtering in the SCIM specification.

> **Note**
>
> The `IdentityStoreUserProvisioner` interface is deprecated. Developers should implement either the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interfaces.

IdentityStoreProvisionerWithFiltering interface implementation

Implement the `IdentityStoreProvisionerWithFiltering` interface to provision and deprovision users and groups to an external user store with list/query and filtering support.

> **Note**
>
> If you do not need to support list/query and filtering, you can implement the `IdentityStoreProvisioner` interface instead.

Implementing this interface requires the following Java packages:

- `com.pingidentity.sdk.provision`
- `com.pingidentity.sdk.provision.exception`
- `com.pingidentity.sdk.provision.users.request`
- `com.pingidentity.sdk.provision.users.response`
- `com.pingidentity.sdk.provision.groups.response`
- `com.pingidentity.sdk.provision.groups.request`

> **Note**
>
> Group support is optional (see Check for group provisioning support).

For each identity store provisioner implementation, in addition to the methods described under Shared plugin interfaces, you must implement the following:

- Create user
- Read user
- Read users (not applicable to the `IdentityStoreProvisioner` interface)
- Update user
- Delete user
- Check for group provisioning support
- Create group
- Read group
- Read groups (not applicable to the `IdentityStoreProvisioner` interface)
- Update group
- Delete group

Create user

```
UserResponseContext createUser(CreateUserRequestContext createRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `createUser()` method of your identity store provisioner in response to create-user requests made to PingFederate services, such as inbound provisioning. This method creates the user in the user store managed by the identity store provisioner.

The `CreateUserRequestContext` contains all information needed to fulfill the request. If the user is successfully provisioned, the method returns a `UserResponseContext` containing the user attributes used to provision the user. The method throws an `IdentityStoreException` if an error occurred during the creation process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Read user

```
UserResponseContext readUser(ReadUserRequestContext readRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `readUser()` method of your identity store provisioner in response to get-user requests made to PingFederate services, such as inbound provisioning. This method retrieves user data from the user store managed by the identity store provisioner.

The `ReadUserRequestContext` contains all information needed to fulfill the request. If the user data is successfully retrieved, the method returns a `UserResponseContext` containing the user attributes for the user. The method throws an `IdentityStoreException` if an error occurred during the retrieval process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Read users

```
UsersResponseContext readUsers(ReadUsersRequestContext readRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `readUsers()` method of your identity store provisioner in response to list/query requests for user attributes made to PingFederate services, such as inbound provisioning. This method retrieves user data from the user store managed by the identity store provisioner.

> **Note**
>
> The `readUsers` method applies only to the `IdentityStoreProvisionerWithFiltering` interface; it does not apply to the `IdentityStoreProvisioner` interface.

The `ReadUsersRequestContext` contains all information needed to fulfill the request. If the user data is successfully retrieved, the method returns a `UsersResponseContext` containing the user attributes satisfying the filter. If an error occurred during the retrieval process, the method returns an `IdentityStoreException`. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Update user

```
UserResponseContext updateUser(UpdateUserRequestContext updateRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `updateUser()` method of your identity store provisioner in response to update-user requests made to PingFederate services, such as inbound provisioning. This method updates the user in the user store managed by the identity store provisioner.

The `UpdateUserRequestContext` contains all information needed to fulfill the request. If the user data is successfully updated, the method returns a `UserResponseContext` containing the user's updated attributes. The method throws an `IdentityStoreException` if an error occurred during the update process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Delete user

```
void deleteUser(DeleteUserRequestContext deleteRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `deleteUser()` method of your identity store provisioner in response to delete-user requests made to PingFederate services, such as inbound provisioning. This method deprovisions the user in the user store managed by the identity store provisioner.

The `DeleteUserRequestContext` contains all information needed to fulfill the request. The method throws an `IdentityStoreException` if an error occurred during the deprovision process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

> **Note**
>
> The plugin implementation can choose not to permanently delete the resource, but must return a `NotFoundException` for all `readUser()`, `updateUser()`, and `deleteUser()` operations associated with the previously deleted ID. In addition, the plugin must not consider the deleted user in conflict calculation. For example, a `createUser()` request for a user with a previously deleted ID should not throw a `ConflictException`.

Check for group provisioning support

```
boolean isGroupProvisioningSupported()
throws IdentityStoreException
```

Implement the `isGroupProvisioningSupported()` method to return true if group provisioning is supported by your identity store provisioner or false otherwise. The method throws an `IdentityStoreException` if an error occurred during the query process. See `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Create group

```
GroupResponseContext createGroup(CreateGroupRequestContext createRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `createGroup()` method of your identity store provisioner in response to create-group requests made to PingFederate services, such as inbound provisioning. This method creates the group in the user store managed by the identity store provisioner if the `isGroupProvisioningSupported()` method returns true; otherwise, it should throw `NotImplementedException`.

The `CreateGroupRequestContext` contains all information needed to fulfill the request, such as group attributes. If the group is successfully provisioned, the method returns a `GroupResponseContext` containing the group attributes used to provision the group. The method throws an `IdentityStoreException` if an error occurred during the creation process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Read group

```
GroupResponseContext readGroup(ReadGroupRequestContext readRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `readGroup()` method of your identity store provisioner in response to get-group requests made to PingFederate services, such as inbound provisioning. This method retrieves group data from the user store managed by the identity store provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `ReadGroupRequestContext` contains all information needed to fulfill the request, such as group ID. If the group data is successfully retrieved, the method returns a `GroupResponseContext` containing the group attributes. The method throws an `IdentityStoreException` if an error occurred during the retrieval process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Read groups

```
GroupsResponseContext readGroups(ReadGroupsRequestContext readRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `readGroups()` method of your identity store provisioner in response to list/query requests for group attributes made to PingFederate services, such as inbound provisioning. This method retrieves group data from the user store managed by the identity store provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

> **Note**
>
> The `readGroups` method applies only to the `IdentityStoreProvisionerWithFiltering` interface; it does not apply to the `IdentityStoreProvisioner` interface.

The `ReadGroupsRequestContext` will contain all information needed to fulfill the request (for example, a filter). If the group data was successfully retrieved, a `GroupsResponseContext` should be returned and contain the group attributes for the groups. An `IdentityStoreException` should be thrown if an error occurred during the retrieval process. See `com.pingidentity.sdk.provision.exception` package for different exceptions that can be thrown.

Update group

```
GroupResponseContext updateGroup(UpdateGroupRequestContext updateRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `updateGroup()` method of your identity store provisioner in response to update-group requests made to PingFederate services, such as inbound provisioning. This method updates the group in the user store managed by the identity store provisioner if the `isGroupProvisioningSupported()` method returns true; otherwise, it should throw `NotImplementedException`.

The `UpdateGroupRequestContext` contains all information needed to fulfill the request, such as group attributes. If the group data is successfully updated, the method returns a `GroupResponseContext` containing the group's updated attributes. The method throws an `IdentityStoreException` if an error occurred during the update process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Delete group

```
void deleteGroup(DeleteGroupRequestContext deleteRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `deleteGroup()` method of your identity store provisioner in response to delete-group requests made to PingFederate services, such as inbound provisioning. This method deprovisions the group in the user store managed by the identity store provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `DeleteGroupRequestContext` contains all information needed to fulfill the request, such as a group ID. The method throws an `IdentityStoreException` if an error occurred during the deprovisioning process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

IdentityStoreUserProvisioner interface implementation

The `IdentityStoreUserProvisioner` interface is deprecated, but you can still implement it to provision and deprovision users to an external user store.

The `IdentityStoreUserProvisioner` interface is deprecated. Developers can implement it to provision and deprovision users, but they should implement either the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interface.

> **Note**
>
> The `IdentityStoreUserProvisioner` interface does not provision or deprovision groups. For group support, see IdentityStoreProvisionerWithFiltering interface implementation.

The following Java packages are required for implementing the interface:

- `com.pingidentity.sdk.provision`
- `com.pingidentity.sdk.provision.exception`
- `com.pingidentity.sdk.provision.users.request`
- `com.pingidentity.sdk.provision.users.response`

For each identity store provisioner implementation, in addition to the methods described under Shared plugin interfaces, you must implement the following methods:

- Create user
- Read user
- Update user
- Delete user

Create user

```
UserResponseContext createUser(CreateUserRequestContext createRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `createUser()` method of your identity store provisioner in response to create-user requests made to PingFederate services, such as inbound provisioning. This method creates the user in the user store managed by the identity store provisioner.

The `CreateUserRequestContext` contains all information needed to fulfill the request. If the user is successfully provisioned, the method returns a `UserResponseContext` containing the user attributes used to provision the user. The method throws an `IdentityStoreException` if an error occurred during the creation process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Read user

```
UserResponseContext readUser(ReadUserRequestContext readRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `readUser()` method of your identity store provisioner in response to get-user requests made to PingFederate services, such as inbound provisioning. This method retrieves user data from the user store managed by the identity store provisioner.

The `ReadUserRequestContext` contains all information needed to fulfill the request. If the user data is successfully retrieved, the method returns a `UserResponseContext` containing the user attributes for the user. The method throws an `IdentityStoreException` if an error occurred during the retrieval process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Update user

```
UserResponseContext updateUser(UpdateUserRequestContext updateRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `updateUser()` method of your identity store provisioner in response to update-user requests made to PingFederate services, such as inbound provisioning. This method updates the user in the user store managed by the identity store provisioner.

The `UpdateUserRequestContext` contains all information needed to fulfill the request. If the user data is successfully updated, the method returns a `UserResponseContext` containing the user's updated attributes. The method throws an `IdentityStoreException` if an error occurred during the update process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

Delete user

```
void deleteUser(DeleteUserRequestContext deleteRequestCtx)
throws IdentityStoreException
```

PingFederate invokes the `deleteUser()` method of your identity store provisioner in response to delete-user requests made to PingFederate services, such as inbound provisioning. This method deprovisions the user in the user store managed by the identity store provisioner.

The `DeleteUserRequestContext` contains all information needed to fulfill the request. The method throws an `IdentityStoreException` if an error occurred during the deprovision process. See the `com.pingidentity.sdk.provision.exception` package for exceptions that can be thrown.

> **Note**
>
> The plugin implementation can choose not to permanently delete the resource, but must return a `NotFoundException` for all `readUser()`, `updateUser()`, and `deleteUser()` operations associated with the previously deleted ID. In addition, the plugin must not consider the deleted user in conflict calculation. For example, a `createUser()` request for a user with a previously deleted ID should not throw a `ConflictException`.

Developing notification publishers

To develop a notification publisher, implement the `NotificationPublisherPlugin` interface.

PingFederate administrators can define and manage notification publishers, as described in Managing notification publisher instances. If those features do not meet your needs, you can develop a custom notification publisher using the PingFederate `NotificationPublisherPlugin` interface and the following Java packages:

- `com.pingidentity.sdk`
- `org.sourceid.saml20.adapter.conf`
- `org.sourceid.saml20.adapter.gui`

The `NotificationPublisherPlugin` interface, which extends the `Plugin` interface, defines the `publishNotification()` method.

For each implementation, define the `publishNotification()` method, in addition to the methods described in Shared plugin interfaces. PingFederate invokes the `publishNotification()` method when publishing a notification. For example, you can configure PingFederate so that an account password change invokes the method.

```
PublishResult publishNotification(String eventType,
  Map<String, String> data,
  Map<String, String> configuration)
```

The method returns the `PublishResult`, the status of the notification that the plugin instance sent.

For more information about the `NotificationPublisherPlugin` interface, see the SDK Javadocs. You can also see a sample implementation in `pingfederate/sdk/plugin-src`.

Building and deploying with Ant

Use the Apache Ant build script to clean, build, package, and deploy projects within the PingFederate Java SDK.

About this task

The PingFederate Java SDK comes with an Apache Ant build script that simplifies building and deploying your projects.

Steps

1. Edit the `build.local.properties` file and set the `target-plugin.name` property to the name of your project subdirectory (see SDK directory structure).

   > **Note**
   >
   > You can develop source code for multiple projects simultaneously, but you can build and deploy only one at a time. Change the value of the `target-plugin.name` property as needed to build and deploy other projects.

2. If your project depends on any third-party `.jar` files, place them into your project's `lib` directory.

   If the directory does not exist, create a new directory called `lib` directly under your project's directory. For example, `pingfederate/sdk/plugin-src/<subproject-name>/lib`.

3. On the command line in the `sdk` directory, use `ant` to clean, build, and package or to build, package, and deploy your project.

Option	Description
Clean the project	`ant clean-plugin`
Compile the project	`ant compile-plugin`
Compile the project and create a `.jar` file	`ant jar-plugin`

> **Note**
>
> The SDK creates a deployment descriptor in the `PF_INF` directory and places it in a `.jar` file. The descriptor tells PingFederate what plugin implementations are contained in the file, and the compiled class files and the deployment descriptor are placed in the `pingfederate/sdk/plugin-src/<subproject-name>/build/classes` directory. The `pf.plugins.<subproject-name>.jar` file is placed in the `pingfederate/sdk/plugin-src/<subproject-name>/build/jar` directory.

To compile the project, create a `.jar` file, and deploy the project to PingFederate, enter:

```
ant deploy-plugin
```

This build target performs the steps described above and deploys any `.jar` files found in the `lib` directory of your subproject.

> **Note**
>
> To deploy your plugin manually to an installation of the PingFederate server, copy the `.jar` file and any third-party `.jar` files into the `/server/default/deploy/` directory of that PingFederate installation.

4. Restart the PingFederate server.

Building and deploying manually

Use a build utility to add directories, create deployment descriptors, and create a `.jar` file to build and deploy your plugins with PingFederate.

Before you begin

To compile your project, you must have the following directories on your classpath:

- `<pf_install>/pingfederate/server/default/lib`
- `<pf_install>/pingfederate/lib`
- `<pf_install>/pingfederate/sdk/lib`
- `<pf_install>/pingfederate/sdk/plugin-src/<subproject-name>/lib`

About this task

To build your project with another build utility, you must create the deployment descriptors for each of your plugins. The deployment descriptor files allow PingFederate to discover your plugins. Once this is complete, use the build tool to create a `.jar` file and deploy it within the appropriate directory.

Steps

1. Add a new directory called `PF-INF` into your project. This directory must be at the root of your `.jar` file, similar to `META-INF`.
2. In `PF-INF`, add an appropriate text file for each type of plugin you created:

Plugin type	File name
IdP Adapter	idp-authn-adapters
SP Adapter	sp-authn-adapters
Custom Data Source	custom-drivers
Token Processor	token-processors
Token Generator	token-generators
Authentication Selector	authentication-selectors
Password Credential Validator	password-credential-validators
Identity Store Provisioner	identity-store-provisioners
CIBA Authenticator	oob-auth-plugins
Notification Publisher	notification-sender

3. In each text file added, specify the fully-qualified class name of each plugin that implements the corresponding plugin interface. Place each class name on a separate line.
4. To create a `.jar`, archive the compiled class files along with the deployment descriptors using your build tool. The deployment descriptors must be in the `PF-INF` directory, located at the root of the `.jar` file.
5. To deploy your plugin, copy the `.jar` file and any third-party `.jar` files into the `<pf_install>/pingfederate/server/default/deploy` directory of the PingFederate installation.

Log messages

You can use a typical logging pattern based on the Apache Commons logging framework to log messages from your adapter, token translator, or custom data source driver.

The service provider (SP) adapter contained in the directory `sdk/adapters-src/sp-adapter-example` shows how to use a logger for your adapter.

Developer’s Reference Guide

This section describes the PingFederate endpoints and APIs.

Use this developer's guide to learn how to develop authentication API-capable adapters and selectors with the following endpoints and APIs:

- OAuth 2.0 endpoints
- Web service interfaces and APIs
- Application endpoints
- Authentication API
- Developing authentication API-capable adapters and selectors

OAuth 2.0 endpoints

When developing OAuth-capable applications, developers must follow the OAuth 2.0 Authorization Framework and OpenID Connect specifications if applicable.

OAuth-capable applications must send requests to various OAuth endpoints to obtain authorization grants, access tokens, refresh tokens, and ID tokens if applicable. Additional endpoints exist for other purposes, including for clients to validate access and refresh tokens, for developers to submit client registrations using the OAuth 2.0 Dynamic Client Registration protocol, and for clients to retrieve metadata about the OpenID Connect and OAuth authorization server configurations.

Each endpoint extends from the runtime server at the base URL. If you configure virtual host names, the endpoints are also accessible at those locations.

Example

If the base URL is www.example.com:9031 and the configured virtual host names are www.example.org and www.example.info, the authorization and token endpoints are accessible at the following locations:

Authorization endpoint /as/authorization.oauth2

- www.example.com:9031/as/authorization.oauth2
- www.example.org:9031/as/authorization.oauth2
- www.example.info:9031/as/authorization.oauth2

Token endpoint /as/token.oauth2

- www.example.com:9031/as/token.oauth2
- www.example.org:9031/as/token.oauth2
- www.example.info:9031/as/token.oauth2s

Related links

- OAuth 2.0 Authorization Framework
- OpenID Connect specifications

Authorization endpoint

The OAuth OAuth authorization server (OAuth AS) uses the authorization endpoint to interact directly with resource owners, authenticate them, and obtain their authorizations.

The OAuth 2.0 Authorization Framework defines the authorization endpoint. Typically, an OAuth client makes an authorization request by directing a resource owner through an HTTP user-agent to the authorization endpoint. After the OAuth AS completes its interaction with the resource owner, the OAuth AS redirects the resource owner's user-agent back to the client's redirect URI with the response to the authorization request.

> **Note**
>
> This endpoint can be used in an OAuth Scope Authentication Selector configuration, which can affect the behavior of the endpoint. For example, the `idp` parameter might be enforced or overridden by policy determined by an instance of the OAuth Scope Authentication Selector.
> This endpoint accepts the HTTP GET and POST methods.

Endpoint: /as/authorization.oauth2

When transmitting through the HTTP POST method, the required `Content-Type` value is `application/x-www-form-urlencoded`. The following table describes parameters for this endpoint.

Parameter	Description
`client_id` (Required)	The client identifier.
`response_mode`	When set to `form_post`, the authorization response is returned to the client in an auto-POST form, in accordance with the OAuth 2.0 Form Post Response Mode specification.
`response_type`	A value of `code` results in the Authorization Code grant type while a value of `token` implies the Implicit grant type. Additionally, a value of `id_token` can be requested by implicit clients. To initiate a Hybrid Flow, multiple response_type values can be specified by space-separating them. When using the Hybrid Flow, some tokens are returned from the Authorization Endpoint and others are returned from the Token Endpoint. For information about multiple-valued response type combinations, see the description of the `restrictedResponseTypes` parameter in OAuth Client Management Service.

Parameter	Description
`code_challenge`	To reduce the risk of authorization code interception attack, supply a one-time string value to associate the authorization request with the token request. For more information, see Proof Key for Code Exchange (PKCE) by OAuth Public Clients. Applicable only when `response_type` parameter value is `code`. Mandatory if the client is required to do so. For more information, see Require Proof Key for Code Exchange (PKCE) in Configuring OAuth clients. **Note** If used, the OAuth client must submit the corresponding code verifier when using the authorization code to obtain an access token. For more information, see `code_verifier` in OAuth grant type parameters.
`code_challenge_method`	Applicable only when the `response_type` parameter value is `code` and a `code_challenge` parameter value is provided. This parameter indicates the transformation method used to derive the `code_challenge` parameter value from that of the `code_verifier` parameter. PingFederate OAuth AS supports two transformation methods: • `plain`, which indicates the `code_challenge` parameter value is that of the `code_verifier` parameter. • `S256`, which indicates the `code_challenge` parameter is derived from the `code_verifier` parameter value as follows `[.parmname]` code_challenge `=Base64Url-encode(SHA256(ASCII([.parmname]` code_verifier `)))`, where: • `ASCII([.parmname]` code_verifier `)` denotes the octets of the ASCII representation of the `code_verifier` value. • `SHA256(octets)` denotes the SHA 256-bit hash of the octets. • `Base64Url-encode(octets)` denotes the base64url encoding of octets; the output is URL-safe. **Note** For detailed information about the transformation method, see Proof Key for Code Exchange (PKCE) by OAuth Public Clients. The `code_challenge_method` parameter value is case-sensitive. An error message is returned to the clients for any other values. Omitting the `code_challenge_method` parameter has the same effect as providing the `code_challenge_method` parameter with a value of `plain`.

Parameter	Description
`redirect_uri`	The URI to which PingFederate redirects the resource owner's user-agent after an authorization is obtained. For OpenID Connect (OIDC) protocol compliance, clients that use the authorization code or implicit grant type must include this parameter in their authorization requests. It is also the default behavior in new PingFederate installations starting with version 9.1.4. For upgraded installations, this requirement remains true for clients that have been configured with more than one redirection URIs. For clients that have been configured with only one redirection URI, this requirement is waived to minimize the impact that it might impose on customers upgrading to version 9.1.4 or a subsequent release. As needed, it can be enabled at a later time. ⓘ **Note** If this parameter is used, the same parameter and value must also be used in subsequent token requests. For more information, see OAuth grant type parameters.
`claims_locales`	Specifies the end-user's preferred languages for claims being returned in a space-separated list, ordered by preference. The values must conform to the IETF BCP 47 guidelines. **Tip** You can map the `claims_locales` value into the persistent grants (and therefore the access tokens, the ID tokens, or both) from an identity provider (IdP) adapter or an IdP connection by selecting **Context** under **Source** and **Requested Claims Locales** under **Value** in the **Contract Fulfillment** tab in the **IdP Adapter Mapping** configuration or the **OAuth Attribute Mapping** configuration in an IdP connection.
`login_hint`	Provides a hint to the PingFederate AS about the end user. For example, when an OAuth client includes a `login_hint` in its authorization request and the authentication source is an HTML Form Adapter instance, the username field in the login form is pre-populated with the `login_hint` parameter value.
`max_age`	Sets an allowable elapsed time in seconds since the end user last authenticated. If the elapsed time exceeds the value of `max_age`, PingFederate prompts the end user for authentication. **Tip** The HTML Form Adapter supports the `max_age` parameter by tracking the authentication time for each user.

`request`	A single, self-contained parameter; a signed JSON Web Token (JWT) whose claims represent the request parameters of the authorization request. The OpenID Connect specification calls this JWT a request object. The `request` parameter is required if a client is configured to transmit request parameters in signed request objects. When PingFederate receives an authorization request, it verifies the digital signature of the signed request object based on the key obtained from the pre-configured JWKS URL or JWKS, and the selected request object signing algorithms. If the signature does not pass the verification process, the request fails. The `request` parameter is optional if a client is not configured to transmit request parameters in signed request objects but is configured with a JWKS URL or an actual JWKS. This flexibility allows the client to transmit request parameters in signed request objects for some requests and without the use of signed request objects for some other transactions.When PingFederate receives an authorization request with a signed request object, it verifies the digital signature of the signed request object based on the key obtained from the pre-configured JWKS URL or JWKS, and the selected request object signing algorithms. If the signature does not pass the verification process, the request fails. If a client is not configured to transmit request parameters in signed request objects and is not configured with a JWKs URL or an actual JWKs, PingFederate ignores the `request` parameter. When PingFederate receives an authorization request with a signed request object, it processes the authorization request and disregards the signed request object. As needed, develop a custom IdP adapter using the PingFederate SDK to extract the `request` parameter and its value from the HTTP request for further processing. PingFederate can decrypt encrypted request objects, which are described in the OpenID Connect 1.0 specification. Request objects with asymmetric encryption must be encrypted using the public keys that PingFederate exposes at `/pf/JWKS`. Request objects with symmetric encryption need a key derived from the client's configured client secret and the client secret must be stored in a reversible format with, for example, symmetric encrypted ID tokens or hash-based message authentication code (HMAC) ID tokens. You can configure PingFederate to accept only request objects that are encrypted by enabling the `front-channel-encryption-required` setting in `jwt-request-object-options.xml`. **Note** If a client includes in an authorization request a request parameter other than `client_id`, as a parameter outside of the signed request object and a claim inside of the signed request object, PingFederate always uses the claim value found inside the signed request object to process the request further. For the `client_id` request parameter, the values outside of the signed request object must match the claim values inside of the signed request object. If the values do not match, PingFederate returns an error message to the client. If a request parameter is found only outside of the signed request object, PingFederate ignores the request parameter and returns no error message. **Tip** Per OAuth and OpenID Connect specifications, a client must always include in an authorization request the `client_id` parameter outside of the signed request object.

Parameter	Description
	For client configuration information, see the Require Signed Request setting in Configuring OAuth clients. For more information about request objects, see RFC 9101: JWT Secured Authorization Request (JAR).
`request_uri`	This parameter indicates that the client is using the pushed authorization requests (PAR) protocol to initiate an authorization flow. The client previously pushed an authorization request payload to the PAR endpoint of the AS. The payload can contain any of the parameters that usually comprise an authorization request and any additional parameters needed for client authentication. After the AS validated the request and saved the payload, it sent the `request_uri` parameter to the client to serve as a reference to the payload. Now the client is using the `request_uri` parameter to request an authorization code or token. The AS uses the value of the `request_uri` to look up the request payload and continue the authorization flow as usual. The AS accepts a particular request URI only once. For more information about PAR, see Pushed authorization requests endpoint.
`scope`	Expresses the scope of the access request as a list of space-separated, case-sensitive strings. For detailed information about scopes, see Scopes and scope management.
`state`	An opaque value used by the client to maintain state between the request and callback. If included, the AS returns this parameter and the given value when redirecting the user agent back to the client.
`ui_locales`	Specifies the end-user's preferred languages for OAuth user interactions in a space-separated list, ordered by preference. The values must conform to the IETF BCP 47 guidelines.
`idp` or `PartnerIdpId`	A PingFederate OAuth AS parameter indicating the entity ID or the connection ID of the IdP with whom to initiate Browser single sign-on (SSO) for user authentication.
`pfidpadapterid` or `IdpAdapterId`	A PingFederate OAuth AS parameter indicating the IdP adapter instance ID of the adapter to use for user authentication. **Note** This parameter might be overridden by policy based on authentication policies. For example, an OAuth Scope Authentication Selector instance could enforce the use of a given adapter instance based on client-requested scopes.
`PolicyAction` (optional)	The HTML Form Adapter immediately returns the value of this parameter in the `policy.action` attribute, allowing the policy to bypass the adapter in favor of an alternative authentication source, provided a rule matching the action is configured. When this parameter is set to `identity.registration` and the adapter is followed by a local identity profile, the user is directed to the registration page for the profile.

If more than one source of authentication is configured in the system and no `pfidpadapterid` or `idp` parameter is provided, PingFederate provides users with an intermediate page asking them to choose among the available sources of authentication. The authentication results in a set of user attributes that must be mapped into the `USER_KEY` attribute for persistent grant storage and the `USER_NAME` attribute that displays on the user authorization page.

OpenID Connect parameters

The following table describes OpenID Connect parameters for this endpoint.

Parameter	Description
`acr_values`	Specifies the Authentication Context Class Reference (acr) values for the AS to use when processing an Authentication Request. Express the values as a space-separated string, and list them in order of preference.
`id_token_hint`	Includes an ID token as a hint to the PingFederate AS about the end user. If the authenticated user does not match the information stored in the ID token, the PingFederate AS rejects the authorization request and returns an error message.
`nonce`	Specifies a string value used to associate a client session with an ID token and to reduce replay attacks. The value passes through unmodified from an authorization request to the ID token.
`prompt`	Specifies whether the AS prompts the end user for reauthentication and consent. Expressed as a list of space-separated, case-sensitive ASCII string values. If included, the client can use this parameter to verify that the end user is still present for the current session or to bring attention to the request. PingFederate supports values of `none`, `login`, and `consent`.

OAuth access token management parameters

PingFederate supports multiple access token management (ATM) instances. Clients can specify an ATM instance by providing the ATM ID (`access_token_manager_id`) or a resource URI (`aud`) in their requests to the PingFederate OAuth AS.

Parameter	Description
`access_token_manager_id`	The `access_token_manager_id` value is the instance ID of the desired ATM instance. When specified, PingFederate uses the desired ATM instance for the request if it is eligible; otherwise it aborts the request. ⓘ **Note** When the `access_token_manager_id` parameter is specified, PingFederate ignores the `aud` parameter.
`aud`	The `aud` is the resource URI the client wants to access. The provided value is matched against resource URIs configured in access token management instances. When a match is found, PingFederate uses the corresponding access token management instance for the request if it is eligible; otherwise it aborts the request.

A match can be an exact match or a partial match where the provided URI has the same scheme and authority parts and a more specific path contained within the path of the pre-configured resource URI. PingFederate takes an exact match over a partial match. If there are multiple partial matches, PingFederate takes the partial match where the provided URI matches more specifically against the pre-configured resource URI.

Example

A resource URI of `https://app.example.local` is a partial match for the following provided URIs:

- `https://app.example.local/file1.ext`
- `https://app.example.local/path/file2.ext`
- `https://app.example.local/path/more`

Client-initiated backchannel authentication endpoint

A CIBA-capable client uses this endpoint to initiate a backchannel, out-of-band flow to authenticate the resource owners and obtain their authorizations.

The OpenID Connect Client Initiated Backchannel Authentication Flow defines the client-initiated backchannel authentication (CIBA) endpoint.

> **Note**
>
> This endpoint accepts only the HTTP POST method.

Endpoint: /as/bc-auth.ciba

The following table describes parameters for this endpoint. The required `Content-Type` value is `application/x-www-form-urlencoded`.

Parameter	Description
`client_id` (Required)	The client identifier. **Important** When sending request parameters of an authentication request with a signed request object, the client must include the `client_id` parameter and its value inside and outside of the `request` parameter value. Both `client_id` parameter values must match.
`scope` (Required)	The scope of the access request. Expressed as a list of space-separated, case-sensitive strings. Scope values are globally defined on the System > OAuth Settings > Scope Management window. You can constrain scopes on a client-to-client basis. This parameter must include the `openid` scope value.

Parameter	Description
`client_notification_token`	A bearer token provided by the client that PingFederate must include when sending a ping callback message to the client's notification endpoint. This usage must conform to the syntax for bearer credentials as defined in section 2.1 of RFC 6750. If the client is configured to use the poll delivery method, this parameter is required.
`id_token_hint`, `login_hint_token`, or `login_hint`	Per the CIBA specification, the client must include one and only one hint for the OpenID Provider to identify the user. The valid hint parameters are `id_token_hint`, `login_hint`, and `login_hint_token`. **`id_token_hint`** Use this parameter to include an ID token as a hint for PingFederate to identify the user. This ID token must be unencrypted. It must be a signed ID token. **`login_hint_token`** Use this parameter to include a JSON web token (JWT) as a hint for PingFederate to identify the user. The attributes of this token can vary from one use case to another. For more information how PingFederate uses the login hint token, see Configuring identity hint contract. **`login_hint`** Use this parameter to provide a hint to PingFederate to identify the user. The value can contain an email address, phone number, account number, subject identifier, username, or any attribute that both sides agreed upon.
`user_code`	A secret code that is known only to the user and verifiable by PingFederate through the use of a Password Credential Validator instance. The purpose of this code is to authorize the transmission of an authentication request to the user's authentication device. If the client record is configured to support user code and associated with a user code-enabled CIBA request policy, this parameter is required.
`binding_message`	An alphanumeric message intended to be made available on both the authentication device and the consumption device. The user can tie them together and decide whether to grant the authorization. When provided, the length of the message must range from 1 - 20 characters.
`requested_expiry`	The requested expiration time of the request in seconds since the generation of the authentication request acknowledgment. **Note** PingFederate honors the requested expiration time only if the value is shorter than that of the **Transaction Lifetime** field found in the associated CIBA request policy.

Parameter	Description
`request`	A single, self-contained parameter; a signed JWT whose claims represent the request parameters of the authentication request. The OpenID Connect specification calls this JWT a request object. The requirement of this parameter and the processing rules vary depending on whether the client is configured with the Require CIBA Signed Requests option. If the client is configured to transmit request parameters to the backchannel authentication endpoint in signed request objects, this parameter is required. In other words, the Require CIBA Signed Requests check box is selected in the configuration of the client. When PingFederate receives an authentication request with a signed request object, it verifies the digital signature of the signed request object based on the key obtained from the pre-configured JWKS URL (or JWKS) and the selected CIBA request object signing algorithm (or algorithms). If the signature does not pass the verification process, the request fails. If a client is not configured to transmit request parameters to the backchannel authentication endpoint in signed request objects, but it is configured with a JWKS URL or an actual JWKS, this parameter is optional. This flexibility allows the client to transmit request parameters in signed request objects for some requests and without the use of signed request objects for some other transactions. When PingFederate receives an authentication request with a signed request object, it verifies the digital signature of the signed request object based on the key obtained from the pre-configured JWKS URL (or JWKS) and the selected CIBA request object signing algorithms. If the signature does not pass the verification process, the request fails. **Important** If the client is not configured to transmit request parameters to the backchannel authentication endpoint in signed request objects, and not configured with a JWKS URL or an actual JWKS, an authentication request with a signed request object will always fail.

Sample authentication request

```
POST /as/bc-auth.ciba HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: www.example.com

client_id=myCibaApp&scope=openid&login_hint=joe@example.com
```

Sample authentication request acknowledgements

- 200 - Success

```
HTTP/1.1 200 OK
...
{
    "auth_req_id": "HQnCgSeUzWNORZEv8n3E8wIip9L3iwBdJAAect04BqdpEsFBGqfxRvoa_Q",
    "interval": 3,
    "expires_in": 120
}
```

- 400 - Bad Request

```
HTTP/1.1 400 Bad Request
...
{
    "error_description": "CIBA authentication requests MUST contain the openid scope value.",
    "error": "invalid_scope"
}
```

```
HTTP/1.1 400 Bad Request
...
{
    "error_description": "Authentication request parameters (such as binding_message) MUST NOT
be present outside of the JWT when a signed authentication request is used.",
    "error": "invalid_request"
}
```

```
HTTP/1.1 400 Bad Request
...
{
    "error_description": "Exactly one (not more, not less) of the hint parameters (i.e.
'login_hint_token', 'id_token_hint' or 'login_hint') must be provided.",
    "error": "invalid_request"
}
```

```
HTTP/1.1 400 Bad Request
...
{
    "error_description": "User could not be sufficiently identified to initiate out-of-band
auth",
    "error": "unknown_user_id"
}
```

```
HTTP/1.1 400 Bad Request
...
{ "error": "invalid_user_code" }
```

```
HTTP/1.1 400 Bad Request
...
{ "error": "missing_user_code" }
```

```
HTTP/1.1 400 Bad Request
...
{
    "error_description": "Client is not configured to support user code but a user_code was
sent in the request.",
    "error": "invalid_request"
}
```

```
HTTP/1.1 400 Bad Request
...
{

    "error_description": "Policy is set to require a token for the user hint but login_hint
was sent.",
    "error": "invalid_request"
}
```

- 401 - Unauthorized

```
HTTP/1.1 401 Unauthorized
...
{
    "error_description": "Invalid client or client credentials.",
    "error": "invalid_client"
}
```

- 500 - Server Error

```
HTTP/1.1 500 Server Error
...
{
    "error_description": "Client is configured to support user code but server policy doesn't
have a PCV configured to do the user code checking",
    "error": "server_error"
}
```

For more information about error responses, see section 13. Authentication Error Response in the specification.

OAuth client identification and authentication

The authentication requirement of this endpoint depends on the client authentication method configured for the clients.

Authentication method	Parameters
Client secret	Clients can present their client identifier and client secret using the HTTP Basic authentication scheme, where the client identifier is the username, and the client secret is the password. Clients can provide credentials using the request parameters `client_id` and `client_secret`. **Important** This is a sensitive parameter. To avoid recording it in web server logs, only pass in this parameter with the HTTP POST method in the message body, or through the HTTP Basic authentication scheme.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject distinguished name (DN) of the client certificate must match values configured for the clients.

Private key JWT or Client Secret JWT	Clients must include request parameters `client_assertion_type` and `client_assertion` in the message body of their requests.
	`client_assertion_type`
	The value describes the format of the assertion as defined by the authorization server. For the private_key_jwt and client_secret_jwt client authentication methods, the value is `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
	`client_assertion`
	The value is the authentication token.
	Example
	`...`
	`client_assertion_type=`
	`urn%3Aietf%3Aparams%3Aoauth%`
	`3Aclient-assertion-type%3Ajwt-bearer&`
	`client_assertion=`
	`eyJhbGciOiJSUzI1NiIs...LbSWi1YO-TILOd4L7ZCg&`
	`...`
	Note
	For readability, line breaks are inserted and the authentication token is truncated.
	Learn more about the private_key_jwt and client_secret_jwt client authentication methods, in Client Authentication and Using Assertions for Client Authentication.

None	Clients must pass in the `client_id` parameter in a query string or the message body to identify themselves.

OAuth access token management parameters

PingFederate supports multiple access token management (ATM) instances. Clients can specify an ATM instance by providing the ATM ID (`access_token_manager_id`) or a resource URI (`aud`) in their requests to the PingFederate OAuth AS.

Parameter	Description
`access_token_manager_id`	The `access_token_manager_id` value is the instance ID of the desired ATM instance. When specified, PingFederate uses the desired ATM instance for the request if it is eligible; otherwise it aborts the request. **Note** When the `access_token_manager_id` parameter is specified, PingFederate ignores the `aud` parameter.
`aud`	The `aud` is the resource URI the client wants to access. The provided value is matched against resource URIs configured in access token management instances. When a match is found, PingFederate uses the corresponding access token management instance for the request if it is eligible; otherwise it aborts the request.

A match can be an exact match or a partial match where the provided URI has the same scheme and authority parts and a more specific path contained within the path of the pre-configured resource URI. PingFederate takes an exact match over a partial match. If there are multiple partial matches, PingFederate takes the partial match where the provided URI matches more specifically against the pre-configured resource URI.

Example

A resource URI of `https://app.example.local` is a partial match for the following provided URIs:

- `https://app.example.local/file1.ext`
- `https://app.example.local/path/file2.ext`
- `https://app.example.local/path/more`

Related links

- Scopes and scope management
- Configuring OAuth clients
- Managing CIBA request policies
- Defining a request policy

Token endpoint

The client presents its authorization grant to the token endpoint to obtain an access token and a refresh token when needed.

The OAuth 2.0 Authorization Framework defines the token endpoint. Because the authorization endpoint directly issues an access token, every authorization grant uses this endpoint except the Implicit grant type.

> **Note**
>
> This endpoint accepts only the HTTP POST method.

Endpoint: /as/token.oauth2

Parameters vary depending on the grant type. Learn more in OAuth grant type parameters. The required `Content-Type` value is `application/x-www-form-urlencoded`.

Like other OAuth 2.0 endpoints, the token endpoint is accessible at the base URL and any configured virtual host names.

If the Token Endpoint Base URL field is configured on the Authorization Server Settings window (System > OAuth Settings > Authorization Server Settings) the token endpoint is also accessible at that location.

For example, if the base URL is https://www.example.com:9031 and the Token Endpoint Base URL value is https://www.example.local:9031, the token endpoints are accessible at the following locations:

- https://www.example.com:9031/as/token.oauth2
- https://www.example.local:9031/as/token.oauth2

OAuth client identification and authentication

The authentication requirement of this endpoint depends on the client authentication method configured for the clients.

Authentication method	Parameters
Client secret	Clients can present their client identifier and client secret using the HTTP Basic authentication scheme, where the client identifier is the username, and the client secret is the password. Clients can provide credentials using the request parameters `client_id` and `client_secret`. **Important** This is a sensitive parameter. To avoid recording it in web server logs, only pass in this parameter with the HTTP POST method in the message body, or through the HTTP Basic authentication scheme.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject distinguished name (DN) of the client certificate must match values configured for the clients.

<table>
<tr><th>Authentication method</th><th>Parameters</th></tr>
<tr><td>Private key JWT or Client Secret JWT</td><td>
<p>Clients must include request parameters <code>client_assertion_type</code> and <code>client_assertion</code> in the message body of their requests.</p>
<p><code>client_assertion_type</code>
The value describes the format of the assertion as defined by the authorization server. For the private_key_jwt and client_secret_jwt client authentication methods, the value is <code>urn:ietf:params:oauth:client-assertion-type:jwt-bearer</code>.</p>
<p><code>client_assertion</code>
The value is the authentication token.</p>
<p>Example</p>
<pre>...
client_assertion_type=
urn%3Aietf%3Aparams%3Aoauth%
3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=
eyJhbGciOiJSUzI1NiIs...LbSWi1YO-TILOd4L7ZCg&
...</pre>
<p><b>Note</b>
For readability, line breaks are inserted and the authentication token is truncated.</p>
<p>Learn more about the private_key_jwt and client_secret_jwt client authentication methods, in Client Authentication and Using Assertions for Client Authentication.</p>
</td></tr>
<tr><td>None</td><td>Clients must pass in the <code>client_id</code> parameter in a query string or the message body to identify themselves.</td></tr>
</table>

OAuth access token management parameters

PingFederate supports multiple access token management (ATM) instances. Clients can specify an ATM instance by providing the ATM ID (`access_token_manager_id`) or a resource URI (`aud`) in their requests to the PingFederate OAuth AS.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td><code>access_token_manager_id</code></td><td>
<p>The <code>access_token_manager_id</code> value is the instance ID of the desired ATM instance. When specified, PingFederate uses the desired ATM instance for the request if it is eligible; otherwise it aborts the request.</p>
<p><b>Note</b>
When the <code>access_token_manager_id</code> parameter is specified, PingFederate ignores the <code>aud</code> parameter.</p>
</td></tr>
</table>

Parameter	Description
`aud`	The `aud` is the resource URI the client wants to access. The provided value is matched against resource URIs configured in access token management instances. When a match is found, PingFederate uses the corresponding access token management instance for the request if it is eligible; otherwise it aborts the request.

A match can be an exact match or a partial match where the provided URI has the same scheme and authority parts and a more specific path contained within the path of the pre-configured resource URI. PingFederate takes an exact match over a partial match. If there are multiple partial matches, PingFederate takes the partial match where the provided URI matches more specifically against the pre-configured resource URI.

Example

A resource URI of `https://app.example.local` is a partial match for the following provided URIs:

- `https://app.example.local/file1.ext`
- `https://app.example.local/path/file2.ext`
- `https://app.example.local/path/more`

OAuth grant type parameters

The `/as/token.oauth2` endpoint accepts other parameters which vary by the grant type presented.

These parameters include OAuth-defined standard parameters and parameters proprietary to PingFederate. The following parameter indicates the grant type of the access token request.

Parameter	Description
`grant_type` (Required)	Indicates the type of grant being presented in exchange for an access token and possibly a refresh token. The value is an extensibility mechanism of the OAuth 2.0 specification. PingFederate supports these values: • `authorization_code` • `refresh_token` • `password` • `client_credentials` • `urn:openid:params:grant-type:ciba` • `urn:ietf:params:oauth:grant-type:device_code` • `urn:ietf:params:oauth:grant-type:jwt-bearer` • `urn:ietf:params:oauth:grant-type:saml2-bearer` • `urn:ietf:params:oauth:grant-type:token-exchange` • `urn:pingidentity.com:oauth2:grant_type:validate_bearer` ⓘ **Note** The following sections define further parameters associated with each grant type.

Authorization code grant type

These parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `authorization_code`.

Parameter	Description
`code` (Required)	The authorization code received from the authorization server during the redirect interaction at the authorization endpoint when the `response_type` parameter is `code`.
`code_verifier`	Required if the authorization request was sent with a `code_challenge` parameter to reduce the risk of code interception attack. If a `code_challenge_method` parameter value is provided in the request for the authorization code, PingFederate OAuth Authorization Server (AS) validates the `code_verifier` parameter value against that of the `code_challenge` value. If the validation returns no errors, PingFederate OAuth AS returns an access token; otherwise it returns an error to the client. For more information about the `code_challenge` parameter, the `code_challenge_method` parameter, and the support for the Proof Key for Code Exchange (PKCE) by OAuth Public Clients specification (https://datatracker.ietf.org/doc/html/rfc7636), see Authorization endpoint.
`redirect_uri`	This parameter is required if the `redirect_uri` parameter was included in the authorization request that resulted in the issuance of the code. For more information, see Authorization endpoint). The value here must match the authorization-request value, if applicable. The parameter is also required for clients with multiple redirection URIs or one redirection URI that uses wildcards. The parameter is optional for clients with only one specific redirection URI.
`scope` (Optional)	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. The requested scope must be equal to or less than the scope originally authorized. If omitted, the scope is treated as equal to the scope originally authorized. Scopes can also be constrained on a client-to-client basis. For more information, see Scopes and scope management.

Refresh token grant type

These parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `refresh_token`.

Parameter	Description
`refresh_token` (Required)	The refresh token issued to the client during a previous access token request. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.

Parameter	Description
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. The requested scope must be equal to or less than the scope originally granted. If omitted, the scope is treated as equal to the original set. Scopes can also be constrained on a client-to-client basis. For more information, see Scopes and scope management.

Resource owner password credentials grant type

These parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `password`.

Parameter	Description
`username` (Required)	The username, encoded as UTF-8.
`password` (Required)	The password, encoded as UTF-8. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can also be constrained on a client-to-client basis. For more information, see Scopes and scope management.
`validator_id`	A PingFederate OAuth AS parameter indicating the instance ID of the password credential validator to check the username and password, and the associated attribute mapping into the `USER_KEY` of the persistent grant. If multiple validator instances are configured and mapped and no `validator_id` parameter is provided, each instance will be tried sequentially, in no particular order, until one succeeds or they all fail.

When a token request triggers an `invalid_grant` error because the corresponding LDAP Username Password Credential Validator instance returns an authentication error, PingFederate includes the relevant message in the error response. See the following example.

```
{
  "error":"invalid_grant",
  "error_description":"We didn't recognize the username or password you entered. Please try again."
}
```

The error description varies based on the error condition that the LDAP Username PCV detects. OAuth-client developers can create custom experiences based on the error messages.

> **Tip**
>
> These customizable messages are stored in the PingFederate message file, `pingfederate-messages.properties`, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.
> You can localize these messages by using the PingFederate localization framework for an international audience. For more information, see Localizing messages for end users.

The `client_id` parameter is not required when the Allow unidentified clients to make resource owner password credentials grants check box is selected in the System > OAuth Settings > Authorization Server Settings window.

Client credentials grant type

The following parameters applies when the `grant_type` parameter for `/as/token.oauth2` is set to `client_credentials`.

Parameter	Description
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can also be constrained on a client-to-client basis. For more information, see Scopes and scope management.

Client-initiated backchannel authentication (CIBA)

The following parameter applies when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:openid:params:grant-type:ciba`.

Parameter	Description
`auth_req_id`	The unique identifier to identify the authentication request.

Sample request

```
POST /as/token.oauth2 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
...

grant_type=urn%3Aopenid%3Aparams%3Agrant-type%3Aciba&client_id=myCibaApp&auth_req_id=yQn...1Vw
```

The `auth_req_id` parameter value in the sample is truncated for readability.

Device authorization grant type

The following parameter applies when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:ietf:params:oauth:grant-type:device_code`.

The OAuth 2.0 Device Authorization Grant specification defines the process that allows a user to grant authorization to a device using a browser on a second device, such as a smart phone or a computer. For more information, see Device authorization grant.

Parameter	Description
`device_code` (Required)	The device code found in the device authorization response.

JWT Bearer Token grant type

These parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:ietf:params:oauth:grant-type:jwt-bearer`.

Parameter	Description
`assertion` (Required)	A JSON Web Token (JWT), as defined in RFC7523, section 2.1. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can be constrained on a client-to-client basis. For more information, see Scopes and scope management.

The `client_id` parameter is not required when the Allow unidentified clients to request extension grants check box is selected in the System > OAuth Settings > Authorization Server Settings window.

SAML 2.0 Bearer Assertion grant type

The following parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:ietf:params:oauth:grant-type:saml2-bearer`.

Parameter	Description
`assertion` (Required)	A single SAML 2.0 assertion, which must be encoded using base64url. For more information, see RFC4648, section 5. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can be constrained on a client-to-client basis. For more information, see Scopes and scope management.

Note

The `client_id` parameter is not required when the **Allow unidentified clients to request extension grants** check box is selected in the **System** > **OAuth Settings** > **Authorization Server Settings** window.

token exchange grant type

The following parameters apply when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:ietf:params:oauth:grant-type:token_exchange`.

Parameter	Description
`resource`	An absolute URI that indicates the target service or resource where the client will use the requested security token. This information lets the authorization server apply the appropriate policy for the target. For example, the authorization server can ensure that the token it issues has the type, content, and encryption that the target requires. If the issued token will be used at multiple targets, multiple `resource` parameters can be used.
`audience`	The logical name of the target where the client will use the requested security token. This serves a purpose like the `resource` parameter. To interpret the audience, the value must be something that both the client and the authorization server understand. The value must be unique within the authorization server. If the issued token will be used at multiple targets, multiple `audience` parameters can be used. When indicating multiple targets, `audience` and `resource` parameters can be used together.
`scope`	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can also be constrained on a client-to-client basis. For more information, see Scopes and scope management.
`requested_token_type`	An identifier for the type of the requested security token. If the requested type is not specified, the authorization server can determine which type is required by the target that the `resource` or `audience` parameter identifies.
`subject_token` (Required)	A security token that represents the identity of the party on behalf of whom the request is being made. Typically, the subject of this token will be the subject of the security token issued in response to the request. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`subject_token_type` (Required)	An identifier for the type of security token in the `subject_token` parameter.
`actor_token`	A security token that represents the identity of the acting party. Typically, this will be the party that is authorized to use the requested security token and act on behalf of the subject. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.

Parameter	Description
`actor_token_type`	An identifier for the type of security token in the `actor_token` parameter. **Important** This parameter is required when the `actor_token` parameter is present in the request but must not be included otherwise.

Access token validation grant type

The following parameter applies when the `grant_type` parameter for `/as/token.oauth2` is set to `urn:pingidentity.com:oauth2:grant_type:validate_bearer`.

Parameter	Description
token (Required)	The bearer access token to be validated. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.

This validation grant type is a custom PingFederate OAuth extension that enables a resource server (RS) to communicate with the OAuth AS while leveraging the established communication and encoding patterns from OAuth 2.0. This grant type allows an RS to verify with the OAuth AS on the validity of a bearer access token that it has received from a client making a protected-resources call.

> **Tip**
>
> An RS client can also use the standard-based Introspection Endpoint at `/as/introspect.oauth2` to validate an access token or a refresh token.

Client authentication is not required. In other words, when creating a client for the sole purpose of validating access tokens, the Client Secret field is optional. For this grant type, the RS acts in the role of a client for the request/response exchange with the OAuth AS to make the validation call.

The response is a standard OAuth access-token response from the token endpoint with some extensions and minor semantic differences in the treatment of some of the parameters. The returned token is in a JSON structure with name-to-value elements or name-to-array elements.

The token type is `urn:pingidentity.com:oauth2:validated_token`, a URN indicating the token represents the attributes associated with the validated access token passed on the request. A `client_id` element is returned indicating the client identifier of the client to whom the grant was made. A `scope` element is returned, if the scope is greater than the default implied scope, indicating the approved scope of the grant. If the issuing access token management (ATM) instance is configured to expand scope groups, the response includes the corresponding sub scopes instead of the scope groups. The `expires_in` element indicates for how many more seconds the token is valid; the value can increase on subsequent validation calls if a token lifetime extension policy is in place.

Sample response when scope group expansion is disabled (the default)

```
{
  "access_token": {
    "Username": "joe",
    "OrgName": "Ping Identity Corporation"
  },
  "scope": "openid AAAGroup",
  "token_type": "urn:pingidentity.com:oauth2:validated_token",
  "expires_in": 7121,
  "client_id": "ac_oic_client"
}
```

`AAAGroup` is a scope group.

Sample response when scope group expansion is enabled

```
{
  "access_token": {
    "Username": "joe",
    "OrgName": "Ping Identity Corporation"
  },
  "scope": "openid AAA1 AAA2",
  "token_type": "urn:pingidentity.com:oauth2:validated_token",
  "expires_in": 7169,
  "client_id": "ac_oic_client"
}
```

`AAA1` and `AAA2` are the expanded outcome of `AAAGroup`.

Validate against all eligible ATM instances

> If multiple ATM instances are eligible, the configuration of the RS client determines whether it must specify the desired ATM instance in its token validation requests. For more information, see Configuring OAuth clients.

After an ATM instance is chosen, PingFederate considers the per-instance session validation settings and processes the validation request. For more information, see Managing session validation settings.

Introspection endpoint

A resource server (RS) client uses the introspection endpoint to validate an access token or a refresh token prior to granting access to a protected-resources call.

The OAuth 2.0 Token Introspection documentation defines the introspection endpoint.

> **Note**
>
> This endpoint accepts only the HTTP POST method.

Endpoint: /as/introspect.oauth2

When transmitting through the HTTP POST method, the required `Content-Type` value is `application/x-www-form-urlencoded`. The RS acts in the role of a client for the request/response exchange with the PingFederate OAuth AS. The validation call is made using the following parameters.

Parameter	Description
`token` (Required)	The bearer access token or refresh token to be validated. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`token_type_hint`	A hint about the type of token submitted for validation. PingFederate supports values of `access_token` and `refresh_token`. Required only when validating a refresh token.

Client authentication is not required; when creating a client for the sole purpose of validating access tokens and refresh tokens, the Client Secret field is optional.

The response is in a JSON structure with a list of name-to-value elements. If a token is valid, the OAuth AS returns to the client a JSON object with the following elements:

- An `"active": true` element to indicate the token is valid. This is also the only element returned by the OAuth AS for a valid refresh token.
- A `client_id` element to indicate the client identifier of the client to whom the grant was made.
- A `scope` element, if the scope is greater than the default implied scope, indicating the approved scope of the grant. If you configured the issuing access token management (ATM) instance to expand scope groups, the response includes the corresponding sub scopes instead of the scope groups.
- An `exp` element indicates the token is valid until the number of seconds since January 1, 1970 UTC (epoch time).
- Other elements from the access token.
- For token introspection requests, if the `Accept` HTTP request header is present the value must be `application/token-introspection+jwt`

  > **Note**
  >
  > To receive the token introspection JSON response as a JSON Web Token (JWT), specify the `Accept` HTTP request header with the value `application/token-introspection+jwt`.

- For token introspection requests, if the `Content-Type` response header is present the value must be `application/token-introspection+jwt`

> **Note**
>
> The response includes the `username` and subject (`sub`) elements only if they were mapped in the ATM instance.

A sample JSON response for a valid access token when a JWT header is present on token introspection requests

```
 {
     "typ": "token-introspection+jwt",
     "alg": "RS256",
     "kid": "wG6D"
  }
```

A sample JSON response for a valid access token. If the JWT Accept header is specified in the request, the JSON will be returned as a JSON web token (JWT).

```
{
     "iss":"https://as.example.com/",
     "aud":"https://rs.example.com/resource",
     "iat":1514797892,
     "token_introspection":
        {
           "active":true,
           "iss":"https://as.example.com/",
           "aud":"https://rs.example.com/resource",
           "iat":1514797822,
           "exp":1514797942,
           "client_id":"paiB2goo0a",
           "scope":"read write dolphin",
           "sub":"Z5O3upPC88QrAjx00dis",
           "birthdate":"1982-02-01",
           "given_name":"John",
           "family_name":"Doe",
           "jti":"t1FoCCaZd4Xv4ORJUWVUeTZfsKhW30CQCrWDDjwXy6w"
        }
   }
```

If a token is invalid, the OAuth AS returns `\{"active": false}` to the client.

A sample response for a valid access token when scope group expansion is disabled (the default)

```
{
  "scope": "openid AAAGroup",
  "active": true,
  "OrgName": "Ping Identity Corporation",
  "token_type": "Bearer",
  "exp": 1556823489,
  "client_id": "ac_oic_client"
}
```

`AAAGroup` is a scope group.

A sample response for a valid access token when scope group expansion is enabled

```
{
  "scope": "openid AAA1 AAA2",
  "active": true,
  "OrgName": "Ping Identity Corporation",
  "token_type": "Bearer",
  "exp": 1556823764,
  "client_id": "ac_oic_client"
}
```

`AAA1` and `AAA2` are the expanded outcome of `AAAGroup`.

Response for a valid refresh token

```
{
  "active": true
  "exp": 1556823764
}
```

> **Note**
>
> If the refresh token is configured to never expire, the `"exp"` attribute will not be returned.

Response for an invalid token

```
{"active":false}
```

OAuth client identification and authentication

The authentication requirement of this endpoint depends on the client authentication method configured for the clients.

Authentication method	Parameters
Client secret	Clients can present their client identifier and client secret using the HTTP Basic authentication scheme, where the client identifier is the username, and the client secret is the password. Clients can provide credentials using the request parameters `client_id` and `client_secret`. **Important** This is a sensitive parameter. To avoid recording it in web server logs, only pass in this parameter with the HTTP POST method in the message body, or through the HTTP Basic authentication scheme.

<table>
<tr><th>Authentication method</th><th>Parameters</th></tr>
<tr><td>Client certificate</td><td>Clients must present their client certificate for mutual TLS authentication. The issuer and the subject distinguished name (DN) of the client certificate must match values configured for the clients.</td></tr>
<tr><td>Private key JWT or Client Secret JWT</td><td>Clients must include request parameters <code>client_assertion_type</code> and <code>client_assertion</code> in the message body of their requests.

<code>client_assertion_type</code>
The value describes the format of the assertion as defined by the authorization server. For the private_key_jwt and client_secret_jwt client authentication methods, the value is <code>urn:ietf:params:oauth:client-assertion-type:jwt-bearer</code>.
<code>client_assertion</code>
The value is the authentication token.

Example
<pre>...
client_assertion_type=
urn%3Aietf%3Aparams%3Aoauth%
3Aclient-assertion-type%3Ajwt-bearer&
client_assertion=
eyJhbGciOiJSUzI1NiIs...LbSWi1YO-TILOd4L7ZCg&
...</pre>Note
For readability, line breaks are inserted and the authentication token is truncated.

Learn more about the private_key_jwt and client_secret_jwt client authentication methods, in Client Authentication and Using Assertions for Client Authentication.</td></tr>
<tr><td>None</td><td>Clients must pass in the <code>client_id</code> parameter in a query string or the message body to identify themselves.</td></tr>
</table>

OAuth access token management parameters

PingFederate supports multiple access token management (ATM) instances. Clients can specify an ATM instance by providing the ATM ID (`access_token_manager_id`) or a resource URI (`aud`) in their requests to the PingFederate OAuth AS.

Parameter	Description
`access_token_manager_id`	The `access_token_manager_id` value is the instance ID of the desired ATM instance. When specified, PingFederate uses the desired ATM instance for the request if it is eligible; otherwise it aborts the request. Note When the `access_token_manager_id` parameter is specified, PingFederate ignores the `aud` parameter.

Parameter	Description
`aud`	The `aud` is the resource URI the client wants to access. The provided value is matched against resource URIs configured in access token management instances. When a match is found, PingFederate uses the corresponding access token management instance for the request if it is eligible; otherwise it aborts the request.

A match can be an exact match or a partial match where the provided URI has the same scheme and authority parts and a more specific path contained within the path of the pre-configured resource URI. PingFederate takes an exact match over a partial match. If there are multiple partial matches, PingFederate takes the partial match where the provided URI matches more specifically against the pre-configured resource URI.

Example

A resource URI of `https://app.example.local` is a partial match for the following provided URIs:

- `https://app.example.local/file1.ext`
- `https://app.example.local/path/file2.ext`
- `https://app.example.local/path/more`

Validate against all eligible ATM instances

> If multiple ATM instances are eligible, the configuration of the RS client determines whether it must specify the desired ATM instance in its token validation requests. For more information, see Configuring OAuth clients.

After an ATM instance is chosen, PingFederate considers the per-instance session validation settings and processes the validation request. For more information, see Managing session validation settings.

Token revocation endpoint

The token revocation endpoint allows clients to notify the authorization server that a previously obtained refresh or access token is no longer needed. The revocation request invalidates the actual token and possibly other tokens based on the same authorization grant.

The OAuth 2.0 Token Revocation documentation defines the token revocation endpoint.

> **Note**
>
> This endpoint accepts only the HTTP POST method.

Endpoint: /as/revoke_token.oauth2

> **Important**
>
> Only Internally Managed Reference Tokens support direct access token revocation. JSON web token (JWT) type access tokens do not support direct revocation. JWT access tokens can only be indirectly revoked if the associated refresh token is revoked, and the JWT's configuration field **Access Grant GUID Claim Name** is set for the given access token manager instance.
> However, you can optionally enable direct revocation for self-contained JWT access tokens assigned to them by enabling JWT access token revocation in access token managers. When enabled, the JWTs require a **Client ID Claim Name** and a minimum **JWT ID Claim Length** of 22 alphanumeric characters. For more information, see the JSON token management tabbed topic and its description of the **Enable Token Revocation** check box in Configuring an access token management instance.

When the authorization server revokes a refresh token, it also revokes the associated access grant and access tokens. When the authorization server revokes an access token, the associated access grant and refresh token remain untouched with the exception of the implicit grant type. If the Reuse Existing Persistent Access Grants for GrantTypes check box is selected in the System > OAuth Settings > Authorization Server Settings window, the implicit access grant will also be revoked with the access token.

The following table describes parameters for this endpoint. The required `Content-Type` value is `application/x-www-form-urlencoded` when transmitting through the HTTP POST method.

Parameter	Description
`token` (Required)	The token that the client wants to revoke. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`token_type_hint`	A hint about the type of token submitted for revocation. PingFederate supports values of `access_token` and `refresh_token`.

The following table describes parameters for this endpoint. The required `Content-Type` value is `application/x-www-form-urlencoded`.

OAuth client identification and authentication

The authentication requirement of this endpoint depends on the client authentication method configured for the clients.

Authentication method	Parameters
Client secret	Clients can present their client identifier and client secret using the HTTP Basic authentication scheme, where the client identifier is the username, and the client secret is the password. Clients can provide credentials using the request parameters `client_id` and `client_secret`. **Important** This is a sensitive parameter. To avoid recording it in web server logs, only pass in this parameter with the HTTP POST method in the message body, or through the HTTP Basic authentication scheme.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject distinguished name (DN) of the client certificate must match values configured for the clients.

Private key JWT or Client Secret JWT	Clients must include request parameters `client_assertion_type` and `client_assertion` in the message body of their requests.
	`client_assertion_type`
	The value describes the format of the assertion as defined by the authorization server. For the private_key_jwt and client_secret_jwt client authentication methods, the value is `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
	`client_assertion`
	The value is the authentication token.
	Example
	`...`
	`client_assertion_type=`
	`urn%3Aietf%3Aparams%3Aoauth%`
	`3Aclient-assertion-type%3Ajwt-bearer&`
	`client_assertion=`
	`eyJhbGciOiJSUzI1NiIs...LbSWi1YO-TILOd4L7ZCg&`
	`...`
	Note
	For readability, line breaks are inserted and the authentication token is truncated.
	Learn more about the private_key_jwt and client_secret_jwt client authentication methods, in Client Authentication and Using Assertions for Client Authentication.

None	Clients must pass in the `client_id` parameter in a query string or the message body to identify themselves.

Grant-management endpoint

Resource owners use the grant-management endpoint to view, and optionally revoke, the persistent access grants they have made.

Two grant-management endpoints are provided. One is for use with parameters. This endpoint is not part of the OAuth specification, but many OAuth providers offer a similar function.

Grants associated with the `USER_KEY` of the authenticated user are displayed. The same attribute mappings from the authentication source to `USER_KEY`, which are used for the authorization endpoint, are used here to look up the user's existing grants.

Endpoints: /as/grants.oauth2 and /as/oauth_access_grants.ping

The following table describes the available parameters for the `/as/grants.oauth2` endpoint. Use only one of them as needed.

Parameter	Description
`idp` or `PartnerIdpId`	Indicates the entity ID of the connection ID of the identity provider (IdP) with whom to initiate browser single sign-on (SSO) for user authentication.
`pfidpadapterid`	Indicates the IdP adapter instance ID of the adapter to use for user authentication. **Note** This parameter may be overridden by policy based on authentication selection configuration. For example, the OAuth Scope Authentication Selector could enforce the use of a given adapter based on client-requested scopes.

If no recent user attributes are found for the session context, the user is redirected to `/as/oauth_access_grants.ping` to initiate the authentication process, which behaves in the same way as the authorization endpoint.

Dynamic client registration endpoint

The client registration endpoint allows developers to dynamically register OAuth clients on a PingFederate authorization server.

The OAuth 2.0 Dynamic Client Registration Protocol defines this endpoint. Developers can send client registrations with the desired properties, such as client metadata, to this endpoint. If the requests are valid, PingFederate evaluates them and returns a response with a client ID and the registered client metadata values.

> **Note**
>
> This runtime endpoint is only active when the dynamic registration client is enabled and configured.

Important

As dynamic client registration can expose your server to unwanted client registrations, we recommend protecting PingFederate by requiring an initial access token, configuring one or more client registration policies, and protecting access to the dynamic client registration endpoint.
You can configure access token requirement and client registration policies using the **System > OAuth Settings > Client Settings** window. To further protect against unauthorized access to the dynamic client registration endpoint, consider using PingAccess or your choice of web access management solution to do so.

Note

This endpoint accepts only the HTTP POST method.

Endpoint: /as/clients.oauth2

Both the request and the response follow the OAuth 2.0 Dynamic Client Registration Protocol.

Example 1

A developer wants to register a client that supports the authorization code flow, two redirection URIs, two scopes, and HTTP Basic as the client authentication method. In this example, PingFederate is not configured to require an initial access token.

Request

```
POST /as/clients.oauth2 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: sso.example.com

{
  "client_name":"Example Org Sample One",
  "redirect_uris":[
    "https://example.org/app1",
    "https://example.org/appM"
  ],
  "scope":"email phone",
  "grant_types":[
    "authorization_code"
  ]
}
```

Response

```
HTTP/1.1 201 Created
Date: Fri, 13 Oct 2017 12:34:56 GMT
Referrer-Policy: origin
Content-Type: application/json
Transfer-Encoding: chunked

{
  "client_id": "dc-F3JxcBlNCtjk36J3Yi4yQK",
  "client_name": "Example Org Sample One",
  "redirect_uris": [
    "https://example.org/app1",
    "https://example.org/appM"
  ],
  "token_endpoint_auth_method": "client_secret_basic",
  "grant_types": [
    "authorization_code"
  ],
  "client_secret": "fYhGUjnkjGp0UPQGaAfdcS",
  "client_secret_expires_at": 0,
  "scope": "phone email",
  "validate_using_all_eligible_atms": false,
  "refresh_token_rolling_policy": "server_default",
  "persistent_grant_expiration_type": "server_default",
  "grant_access_session_revocation_api": false
  "grant_access_session_management_api": false
}
```

PingFederate returns `201 Created`, the client ID, and other registered client metadata after creating the new client.

Additionally, when a registration request does not specify a client authentication method (`token_endpoint_auth_method`), PingFederate defaults to `client_secret_basic` per OAuth 2.0 Dynamic Client Registration Protocol.

Example 2

A developer wants to register a client that supports the authorization code flow, refresh tokens, one redirection URI, one `profile` scope, and HTTP Basic as the client authentication method. In this example, PingFederate is not configured to require an initial access token. However, the `profile` scope is restricted. As a result, the registration request should fail.

Request

```
POST /as/clients.oauth2 HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: www.example.com

{
  "client_name":"Example Org Sample Two",
  "redirect_uris":[
    "https://example.org/app2"
  ],
  "scope":"profile",
  "grant_types":[
    "authorization_code",
    "refresh_token"
  ]
}
```

Response

```
HTTP/1.1 400 Bad Request
Date: Fri, 13 Oct 2017 13:00:00 GMT
Referrer-Policy: origin
Content-Type: application/json
Transfer-Encoding: chunked
{
  "error": "invalid_client_metadata",
  "error_description": "The requested scope is invalid."
}
```

PingFederate returns `400 Bad Request` and the relevant error message when a client registration fails.

Example 3

A developer wants to register a client that supports the authorization code flow, two redirection URIs, two scopes, and HTTP Basic as the client authentication method. In this example, PingFederate is configured to require an initial access token.

Request

```
POST /as/clients.oauth2 HTTP/1.1
Content-Type: application/json
Accept: application/json
Authorization: Bearer
eyJhbGciOiJSUzI1NiIsImtpZCI6ImsxIn0.eyJzY29wZSI6WyJkQ1IiXSwiY2xpZW50X2lkX25hbWUiOiJwYXJ0bmVyRGV2X0FjbWUiLCJhZ2lkIjoiM
CHtcQ79Wefz2Sw5GOB5LfV9mWJ0n3vzJ93Ie7wbEAkalIFg53J-9e7s59MjA1igx6ybflGMQ9QAjYobs-
jM24arJZZgopEXvcx6IQpyU8U4AMTJ7tr9Lmody8P0QZOKcUDBTT5egv9vr5NuXCtUBfVPhGZ-3p5g5mwrnGHBfqZOAsg7U4hKq8cauKQtVyBBV9iIZN
oDhmilbmiga4319YSFfX5-
U3li9XPeN3JZB2ukLbTFjjVIVLJIInbSR_IFTWP5Irg92aXLrIfm5MvBp8D1fOU6xYjbgjvw9QKNiFFVD7oEeJG9MwzgcGruw
Host: www.example.com

{
  "client_name":"Example Org Sample Three",
  "redirect_uris":[
    "https://example.org/app3",
    "https://example.org/appN"
  ],
  "scope":"email phone",
  "grant_types":[
    "authorization_code"
  ]
}
```

Response

```
HTTP/1.1 201 Created
Date: Fri, 13 Oct 2017 15:30:00 GMT
Referrer-Policy: origin
Content-Type: application/json
Transfer-Encoding: chunked

{
  "client_id": "dc-rqUtii4vRXj5NMztkAeJ1S",
  "client_name": "Example Org Sample Three",
  "redirect_uris": [
    "https://example.org/app3",
    "https://example.org/appN"
  ],
  "token_endpoint_auth_method": "client_secret_basic",
  "grant_types": [
    "authorization_code"
  ],
  "client_secret": "p7MD0Ul1DNI9xRDc5kcOxs",
  "client_secret_expires_at": 0,
  "scope": "phone email",
  "validate_using_all_eligible_atms": false,
  "refresh_token_rolling_policy": "server_default",
  "persistent_grant_expiration_type": "server_default",
  "grant_access_session_revocation_api": false
  "grant_access_session_management_api": false
}
```

The registration request must include an `Authorization` HTTP header with a valid access token as its value.

If the authorization fails, PingFederate returns the following JSON payload in the response.

```
{
  "error": "invalid_access_token",
  "error_description": "Please provide a valid Access Token with the correct scope"
}
```

Related links

- Configuring dynamic client registration settings
- Supported client metadata

Device authorization endpoint

The device authorization endpoint allows a user to grant authorization to a device client using a browser on a second device, such as a smart phone or a computer.

The OAuth 2.0 Device Authorization Grant defines the device authorization endpoint. Based on the specification, the device sends a device authorization request to PingFederate, the authorization server (AS), at its device authorization endpoint.

Note

Per OAuth specifications, this endpoint accepts only the HTTP POST method.

Endpoint: /as/device_authz.oauth2

The required `Content-Type` value is `application/x-www-form-urlencoded`. The following table describes parameters for this endpoint.

Parameter	Description
`client_id`	A unique identifier the client provides to the resource server to identify itself. This identifier is included with every request the client makes
`scope` (Optional)	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scopes can also be constrained on a client-to-client basis. For more information about scopes, see Scopes and scope management.

Both the request and the response follow the OAuth 2.0 Device Authorization Grant.

Example

Example request

```
POST /as/device_authz.oauth2 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
...

client_id=df_client
```

Example

Response codes and example responses

200 - Success

```
HTTP/1.1 200 OK

...
{
    "user_code": "YYD6-CD4T",
    "device_code": "4EHsIngavzIPvvqMlFgQlseTCsH7EpU75f9yGvj60T",
    "interval": 5,
    "verification_uri_complete": "https://www.example.com/as/user_authz.oauth2?user_code=YYD6-CD4T",
    "verification_uri": "https://www.example.com/as/user_authz.oauth2",
    "expires_in": 600
}
```

400 - Bad Request

```
HTTP/1.1 400 Bad Request

...
{
    "error_description": "The requested scope(s) must be blank or a subset of the provided scopes.",
    "error": "invalid_scope"
}
```

401 - Unauthorized

```
HTTP/1.1 401 Unauthorized

...
{
    "error_description": "Invalid client or client credentials.",
    "error": "invalid_client"
}
```

OAuth client identification and authentication

The authentication requirement of this endpoint depends on the client authentication method configured for the clients.

Authentication method	Parameters
Client secret	Clients can present their client identifier and client secret using the HTTP Basic authentication scheme, where the client identifier is the username, and the client secret is the password. Clients can provide credentials using the request parameters `client_id` and `client_secret`. **Important** This is a sensitive parameter. To avoid recording it in web server logs, only pass in this parameter with the HTTP POST method in the message body, or through the HTTP Basic authentication scheme.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject distinguished name (DN) of the client certificate must match values configured for the clients.

Authentication method	Parameters

Private key JWT or Client Secret JWT	Clients must include request parameters `client_assertion_type` and `client_assertion` in the message body of their requests.
	`client_assertion_type`
	The value describes the format of the assertion as defined by the authorization server. For the private_key_jwt and client_secret_jwt client authentication methods, the value is `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
	`client_assertion`
	The value is the authentication token.
	Example
	`...`
	`client_assertion_type=`
	`urn%3Aietf%3Aparams%3Aoauth%`
	`3Aclient-assertion-type%3Ajwt-bearer&`
	`client_assertion=`
	`eyJhbGciOiJSUzI1NiIs...LbSWi1YO-TILOd4L7ZCg&`
	`...`
	Note
	For readability, line breaks are inserted and the authentication token is truncated.
	Learn more about the private_key_jwt and client_secret_jwt client authentication methods, in Client Authentication and Using Assertions for Client Authentication.

None	Clients must pass in the `client_id` parameter in a query string or the message body to identify themselves.

Related links

- Device authorization grant
- Configuring authorization server settings

User authorization endpoint

The user authorization endpoint allows a user to grant authorization to a device client using a browser on a second device, such as a smart phone or a computer.

Based on the OAuth 2.0 Device Authorization Grant specification, the user goes to the user authorization endpoint of the PingFederate authorization server (AS) to complete the authorization process.

> **Note**
>
> This endpoint accepts the HTTP GET and POST methods.

Endpoint: /as/user_authz.oauth2

The following table describes parameter for this endpoint. The required `Content-Type` value is `application/x-www-form-urlencoded` when transmitting through the HTTP POST method.

Parameter	Description
`user_code` (Optional)	This value represents the activation code.

Both the request and the response follow the OAuth 2.0 Device Authorization Grant.

Example

Example request

```
POST /as/user_authz.oauth2 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
...

user_code=QQWP-TJ6B
```

Subsequent responses

Phase 1: Activation code verification

If the verification request does not include an activation code, PingFederate returns the Connect a device (user code prompt) page, prompting the user to enter the activation code shown by the device.If the verification request includes an activation code, PingFederate returns the Connect a device (pre-populated user code prompt) page, prompting the user to confirm the activation code from the verification request matches the activation code shown by the device. PingFederate skips this step if the Bypass Activation Code Confirmation option is enabled globally or individually for that invoking client.PingFederate validates the activation code, prompts the user to enter another activation code if it is invalid, or moves to the next phase.

Phase 2: Authentication

PingFederate prompts the user to fulfill the authentication requirements based on OAuth grant mapping configurations and authentication policies.If the user fulfills the authentication requirements, PingFederate moves to the next phase; otherwise it returns an error message to the user.

Phase 3: Authorization

PingFederate returns the Request for Approval page, prompting the user to approve or deny the requested scopes. PingFederate skips this step if the Bypass Authorization Approval option is enabled globally or individually for that invoking client and the user has granted authorization for the requested scopes previously.PingFederate returns the Connect a device (result) page to the user. The message reflects the authorization status.If the user approves the requested scopes, the next time the device sends a device access token request to PingFederate at its token endpoint, PingFederate returns an access token to the device.When an error occurs, PingFederate returns `400 Bad Request` in response to the device access token request.

Example of 400 Bad Request

```
HTTP/1.1 400 Bad Request

...
{"error_description":"Authorization request is denied","error":"access_denied"}
```

```
HTTP/1.1 400 Bad Request

...
{"error_description":"Device code not found, expired or invalid","error":"invalid_grant"}
```

```
HTTP/1.1 400 Bad Request

...
{"error_description":"The authorization request has expired.","error":"expired_token"}
```

Related links

- Device authorization grant
- Configuring authorization server settings
- OAuth user-facing pages

OpenID Provider configuration endpoint

The OpenID Provider (OP) configuration endpoint provides configuration information for the OAuth clients to interface with PingFederate using the OpenID Connect protocol.

This endpoint returns configuration information that is controlled by a template file and can be customized to suit multiple use cases simultaneously.

This public endpoint accepts HTTP GET requests without authentication.

Endpoint: /.well-known/openid-configuration

The following table describes the parameter for this endpoint.

Parameter	Description
`policy_id`	Indicates the OpenID Connect policy from which PingFederate derives the attributes to include under `claims_supported` in the response body. If omitted, PingFederate includes the attributes based on the default policy.

Example

Example response

```
$ curl -s https://localhost:9031/.well-known/openid-configuration|python -m json.tool
{
    "authorization_endpoint": "https://localhost:9031/as/authorization.oauth2",
    "backchannel_authentication_endpoint": "https://localhost:9031/as/bc-auth.ciba",
    "backchannel_authentication_request_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "backchannel_token_delivery_modes_supported": [
        "poll",
        "ping"
    ],
    "backchannel_user_code_parameter_supported": true,
    "code_challenge_methods_supported": [
        "plain",
        "S256"
    ],
    "claim_types_supported": [
        "normal"
    ],
    "claims_parameter_supported": false,
    "claims_supported": [
        "address",
        "birthdate",
        "email",
        "email_verified",
        "family_name",
        "gender",
        "given_name",
        "locale",
        "middle_name",
        "name",
        "nickname",
        "phone_number",
        "phone_number_verified",
        "picture",
        "preferred_username",
        "profile",
        "sub",
        "updated_at",
        "website",
        "zoneinfo"
    ],
    "device_authorization_endpoint": "https://localhost:9031/as/device_authz.oauth2",
    "grant_types_supported": [
        "implicit",
        "authorization_code",
        "refresh_token",
        "password",
        "client_credentials",
        "urn:pingidentity.com:oauth2:grant_type:validate_bearer",
        "urn:ietf:params:oauth:grant-type:jwt-bearer",
        "urn:ietf:params:oauth:grant-type:saml2-bearer",
        "urn:ietf:params:oauth:grant-type:device_code",
        "urn:openid:params:grant-type:ciba"
    ],
    "id_token_encryption_alg_values_supported": [
        "dir",
        "A128KW",
        "A192KW",
        "A256KW",
        "A128GCMKW",
        "A192GCMKW",
        "A256GCMKW",
        "ECDH-ES",
        "ECDH-ES+A128KW",
        "ECDH-ES+A192KW",
        "ECDH-ES+A256KW",
        "RSA-OAEP"
    ],
    "id_token_encryption_enc_values_supported": [
        "A128CBC-HS256",
        "A192CBC-HS384",
        "A256CBC-HS512",
        "A128GCM",
        "A192GCM",
        "A256GCM"
    ],
    "id_token_signing_alg_values_supported": [
        "none",
        "HS256",
        "HS384",
        "HS512",
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "introspection_endpoint": "https://localhost:9031/as/introspect.oauth2",
    "issuer": "https://localhost:9031",
    "jwks_uri": "https://localhost:9031/pf/JWKS",
    "ping_end_session_endpoint": "https://localhost:9031/idp/startSLO.ping",
    "ping_revoked_sris_endpoint": "https://localhost:9031/pf-ws/rest/sessionMgmt/revokedSris",
    "registration_endpoint": "https://localhost:9031/as/clients.oauth2",
    "request_object_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "request_parameter_supported": true,
    "request_uri_parameter_supported": false,
    "response_modes_supported": [
        "fragment",
        "query",
        "form_post"
    ],
    "response_types_supported": [
        "code",
        "token",
        "id_token",
        "code token",
        "code id_token",
        "token id_token",
        "code token id_token"
    ],
    "revocation_endpoint": "https://localhost:9031/as/revoke_token.oauth2",
    "scopes_supported": [
        "address",
        "phone",
        "edit",
        "openid",
        "profile",
        "admin",
        "email"
    ],
    "subject_types_supported": [
        "public",
        "pairwise"
    ],
    "token_endpoint": "https://www.example.com:9031/as/token.oauth2",
    "token_endpoint_auth_methods_supported": [
        "client_secret_basic",
        "client_secret_post",
        "private_key_jwt",
        "client_secret_jwt"
    ],
    "token_endpoint_auth_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "HS256",
        "HS384",
        "HS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "userinfo_endpoint": "https://localhost:9031/idp/userinfo.openid"
}
```

Notable metadata parameters

CIBA user code support

The `backchannel_user_code_parameter_supported` parameter indicates whether the default CIBA request policy supports user codes, which are an optional feature in the CIBA specification.In the previous example, because the User Code PCV field is configured with a Password Credential Validator instance in the default CIBA request policy, the value of the `backchannel_user_code_parameter_supported` parameter is `true`. For more information, see OpenID Connect Client Initiated Backchannel Authentication Flow and Defining a request policy.

Digital signature algorithms

The `backchannel_authentication_request_signing_alg_values_supported`, `id_token_signing_alg_values_supported`, `token_endpoint_auth_signing_alg_values_supported`, and `request_object_signing_alg_values_supported` parameters provide lists of supported algorithms to process digital signatures.In this example, because PingFederate is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, the endpoint includes additional RSASSA-PSS digital signature algorithms (`PS256`, `PS384`, and `PS512`) in its response. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. Deploying PingFederate to run on a Java 8 or a Java 11 environment will have the same result.

JWKS endpoint

The JWKS endpoint, `jwks_uri`, returns a set of public keys for OAuth and OpenID Connect. Clients can use this information to verify the integrity of asymmetrically-signed ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects.

Scopes

The OP configuration endpoint returns all common static scopes and common scope groups but not exclusive static scopes, exclusive scope groups, common dynamic scopes, or exclusive dynamic scopes by default. The response can be customized by editing a template file to include or exclude individual scopes and scope groups.

Token endpoint

The token endpoint, `token_endpoint`, is used by clients to obtain access tokens and refresh tokens if applicable.In the previous example, because the Token Endpoint Base URL is set to `https://www.example.com:9031` in the System > OAuth Settings > Authorization Server Settings window, the `token_endpoint` value is set to `https://www.example.com:9031/as/token.oauth2`. For more information, see Configuring authorization server settings and Token endpoint.

Related links

- OpenID Connect Discovery 1.0 (openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata)
- Configuring OpenID Connect policies
- Customizing a configuration endpoint response

OAuth authorization server metadata endpoint

The OAuth authorization server metadata endpoint provides configuration information that OAuth clients need to interface with PingFederate using the OAuth 2.0 protocol.

PingFederate supports OAuth 2.0 authorization server metadata (https://www.rfc-editor.org/rfc/rfc8414). This lets OAuth clients retrieve relevant endpoints and other details about features that PingFederate supports.

The information returned by the OAuth authorization server metadata endpoint is controlled by the `openid-configuration.template.json` template file. You can customize that file to suit OAuth and OpenID Connect use cases simultaneously.

Endpoint: /.well-known/oauth-authorization-server

The OAuth authorization server metadata endpoint is `/.well-known/oauth-authorization-server`. This public endpoint accepts HTTP GET requests without authentication.

When PingFederate is configured with a virtual issuer and a path, the patterns for the OpenID Connect and OAuth authorization server metadata endpoints differ from one another because they are based on different specifications. The OAuth 2.0 metadata specification requires the issuer path to be after `/.well-known/oauth-authorization-server`. The OpenID Connect metadata specification requires the issuer path to be before `/.well-known/openid-configuration`.

For example, if the virtual issuer's host name is `sso.example.com` and the path is `/issuer1`, which produces `/https://sso.example.com/issuer1`, then:

- the OAuth authorization server metadata endpoint is

  ```
  https://sso.example.com/.well-known/oauth-authorization-server/issuer1
  ```

- the OpenID Connect metadata endpoint is

  ```
  https://sso.example.com/issuer1/.well-known/openid-configuration
  ```

The response from the OAuth authorization server metadata endpoint is like the response from the OpenID Connect metadata endpoint but doesn't include details specific to OpenID Connect. The following code block shows an example of a response from the OAuth authorization server metadata endpoint.

ExampleExample response

```
$ curl -s https://localhost:9031/.well-known/oauth-authorization-server|python -m json.tool
{
    "authorization_endpoint": "https://localhost:9031/as/authorization.oauth2",
    "backchannel_authentication_endpoint": "https://localhost:9031/as/bc-auth.ciba",
    "backchannel_authentication_request_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "backchannel_token_delivery_modes_supported": [
        "poll",
        "ping"
    ],
    "backchannel_user_code_parameter_supported": true,
    "code_challenge_methods_supported": [
        "plain",
        "S256"
    ],
    "claims_supported": [
        "address",
        "birthdate",
        "email",
        "email_verified",
        "family_name",
        "gender",
        "given_name",
        "locale",
        "middle_name",
        "name",
        "nickname",
        "phone_number",
        "phone_number_verified",
        "picture",
        "preferred_username",
        "profile",
        "sub",
        "updated_at",
        "website",
        "zoneinfo"
    ],
    "device_authorization_endpoint": "https://localhost:9031/as/device_authz.oauth2",
    "grant_types_supported": [
        "implicit",
        "authorization_code",
        "refresh_token",
        "password",
        "client_credentials",
        "urn:pingidentity.com:oauth2:grant_type:validate_bearer",
        "urn:ietf:params:oauth:grant-type:jwt-bearer",
        "urn:ietf:params:oauth:grant-type:saml2-bearer",
        "urn:ietf:params:oauth:grant-type:device_code",
        "urn:openid:params:grant-type:ciba"
    ],
    "introspection_endpoint": "https://localhost:9031/as/introspect.oauth2",
    "issuer": "https://localhost:9031",
    "jwks_uri": "https://localhost:9031/pf/JWKS",
    "ping_end_session_endpoint": "https://localhost:9031/idp/startSLO.ping",
    "ping_revoked_sris_endpoint": "https://localhost:9031/pf-ws/rest/sessionMgmt/revokedSris",
    "registration_endpoint": "https://localhost:9031/as/clients.oauth2",
    "request_object_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "request_parameter_supported": true,
    "request_uri_parameter_supported": false,
    "response_modes_supported": [
        "fragment",
        "query",
        "form_post"
    ],
    "response_types_supported": [
        "code",
        "token",
        "id_token",
        "code token",
        "code id_token",
        "token id_token",
        "code token id_token"
    ],
    "revocation_endpoint": "https://localhost:9031/as/revoke_token.oauth2",
    "scopes_supported": [
        "address",
        "phone",
        "edit",
        "openid",
        "profile",
        "admin",
        "email"
    ],
    "token_endpoint": "https://www.example.com:9031/as/token.oauth2",
    "token_endpoint_auth_methods_supported": [
        "client_secret_basic",
        "client_secret_post",
        "private_key_jwt"
    ],
    "token_endpoint_auth_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ]
}
```

Notable metadata parameters

CIBA user code support

The `backchannel_user_code_parameter_supported` parameter indicates whether the default CIBA request policy supports user codes, which are an optional feature in the CIBA specification.In the previous example, because the User Code PCV field is configured with a Password Credential Validator instance in the default CIBA request policy, the value of the `backchannel_user_code_parameter_supported` parameter is `true`. For more information, see OpenID Connect Client Initiated Backchannel Authentication Flow and Defining a request policy.

Digital signature algorithms

The `backchannel_authentication_request_signing_alg_values_supported`, `token_endpoint_auth_signing_alg_values_supported`, and `request_object_signing_alg_values_supported` parameters provide lists of supported algorithms to process digital signatures.In this example, because PingFederate is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, the endpoint includes additional RSASSA-PSS digital signature algorithms (`PS256`, `PS384`, and `PS512`) in its response. For more information on HSM integration and static keys, see Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. Deploying PingFederate to run on a Java 8 or a Java 11 environment will have the same result.

JWKS endpoint

The JWKS endpoint, `jwks_uri`, returns a set of public keys for OAuth and OpenID Connect. Clients can use this information to verify the integrity of asymmetrically-signed ID tokens, JSON web tokens (JWTs) for client authentication, and OpenID Connect request objects.

Scopes

The OP configuration endpoint returns all common static scopes and common scope groups but not exclusive static scopes, exclusive scope groups, common dynamic scopes, or exclusive dynamic scopes by default. The response can be customized by editing a template file to include or exclude individual scopes and scope groups.

Token endpoint

The token endpoint, `token_endpoint`, is used by clients to obtain access tokens and refresh tokens if applicable.In the previous example, because the Token Endpoint Base URL is set to `https://www.example.com:9031` in the System > OAuth Settings > Authorization Server Settings window, the `token_endpoint` value is set to https://www.example.com:9031/as/token.oauth2. For more information, see Configuring authorization server settings and Token endpoint.

UserInfo endpoint

OAuth clients can present access tokens to the `UserInfo` endpoint to retrieve additional information about the resource owners.

You can customize the amount of information presented by the endpoint by using OpenID Connect policies. Information can include specification-defined attributes (standard attributes) and non-standard attributes. Scopes, authorized by the users, also determine the attributes to be returned.

This endpoint accepts HTTP GET requests without parameters. Clients must present valid access tokens for authentication.

Example

UserInfo endpoint /idp/userinfo.openid

The following example shows a request to `UserInfo` endpoint `/idp/userinfo.openid` and the response. The self-contained access token in the request's Authorization header is truncated for readability.

```
$ curl -s https://localhost:9031/idp/userinfo.openid -H 'Authorization: Bearer eyJ...9-g'|python -m json.tool

{
    "email": "auser@example.com",
    "phone_number": "(555) 555-5555",
    "phone_number_verified": true,
    "sub": "joe"
}
```

If the access token presented is not valid, PingFederate returns `401 Unauthorized`.

> **Note**
>
> To use a demonstrating proof-of-possession (DPoP)-bound access token, use the DPoP authentication scheme in the Authorization header and include a DPoP Proof JWT in the DPoP header. Learn more in the OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer specification and the description of the PingFederate DPoP settings in Configuring authorization server settings.

Self-contained tokens

If clients using self-contained access tokens are expected to contact the `UserInfo` endpoint, consider the following implications:

Client ID Claim Name

This field's default value is `client_id`. When this field is configured with a value, PingFederate includes the client ID of the requesting client as a claim in the self-contained tokens. The claim name is the value of the Client ID Claim Name field.

If the field value is empty, PingFederate will not include the client ID of the requesting client in the self-contained tokens. In this scenario, the access token manager (ATM) instance used by the default OpenID Connect policy must remain accessible to all clients, or clients using self-contained access tokens issued by this ATM instance will not be able to retrieve additional claims from the UserInfo endpoint. Instead, they receive an HTTP status code `401 Unauthorized` from PingFederate. Learn more in Defining access control.

Learn more about the Client ID Claim Name in Configuring an access token management instance.

Scope Claim Name

This field's default value is `scope`. When this field is configured with a value, PingFederate includes the requested scopes as a claim in the self-contained tokens. The claim name is the value of the Scope Claim Name field.

If the field value is empty, PingFederate will not include any scope information in the self-contained token, and clients using self-contained access tokens issued by this ATM instance will not be able to retrieve additional claims from the UserInfo endpoint. Instead, they receive an HTTP status code `403 Forbidden` from PingFederate.

Learn more about the Scope Claim Name in Configuring an access token management instance.

Related links

- Token models and management
- Configuring an access token management instance
- Configuring OpenID Connect policies

Pushed authorization requests endpoint

The PingFederate authorization server (AS) can provide a pushed authorization requests (PAR) endpoint `/as/par.oauth2`. OAuth 2.0 clients can use the PAR endpoint to securely *initiate* authorization flows.

When the PAR endpoint is enabled, a client can push an authorization request payload to the AS with a direct back-channel request. This is a more secure method of sending sensitive data, such as personally identifiable information, than sending it with a browser on the front channel. The payload contains parameters that are application/x-www-form-urlencoded formatted. The PAR endpoint can accept all parameters that usually comprise an authorization request and any additional parameters needed for client authentication. It also can accept signed requests.

> **Note**
>
> The PAR endpoint only accepts the HTTP POST method.

After the AS validates the request and saves the payload, it returns the `request_uri` parameter to serve as a reference to the payload. The response also indicates the lifetime of the request URI. The default lifetime is 60 seconds.

Sample request for a request_uri to the PAR endpoint

```
POST /as/par.oauth2 HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0Mzo3RmpmcDBaQnIxS3REUmJuZlZkbUl3

response_type=code
&client_id=s6BhdRkqt3
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Sample response with a request_uri from the PAR endpoint

```
HTTP/1.1 201 Created
Cache-Control: no-cache, no-store
Content-Type: application/json

{
  "expires_in": 60,
  "request_uri": "urn:example:bwc4JK-ESC0w8acc191e-Y1LTC2"
}
```

Subsequently, the client uses the front channel to request an authorization code or token, sending the `request_uri` parameter to the AS authorization endpoint. The AS uses the request URI to look up the request payload and continue the authorization flow as usual. The AS accepts a particular request URI only once.

Sample request for an authorization code or token to the authorization endpoint

```
GET /as/authorization.oauth2?client_id=s6BhdRkqt3
  &request_uri=urn%3Aexample%3Abwc4JK-ESC0w8acc191e-Y1LTC2 HTTP/1.1
Host: as.example.com
```

To let OAuth clients use the PAR endpoint, you must enable PAR on the AS. Additionally, you can configure individual clients so that they must use the PAR endpoint to initiate authorization flows. For more information about PAR settings, see Configuring authorization server settings and Configuring OAuth clients.

When PAR is enabled on the AS, the well-known endpoint `/.well-known/openid-configuration` includes these PAR parameters in responses:

- `pushed_authorization_request_endpoint` contains the PAR endpoint
- `require_pushed_authorization_requests` indicates whether the AS requires clients to use PAR

PingFederate can decrypt encrypted request objects, which are described in the OpenID Connect 1.0 specification. Request objects with asymmetric encryption must be encrypted using the public keys that PingFederate exposes at `/pf/JWKS`. Request objects with symmetric encryption need a key derived from the client's configured client secret and the client secret must be stored in a reversible format with, for example, symmetric encrypted ID tokens or hash-based message authentication code (HMAC) ID tokens. You can configure PingFederate to accept only request objects that are encrypted by enabling the `front-channel-encryption-required` setting in `jwt-request-object-options.xml`.

You can use parameters from PAR payloads in authentication policies. The policies can reference them like other tracked HTTP request parameters. Make PAR parameters available to policies by adding them to the Policies window's Tracked HTTP Parameters tab.

> **Note**
>
> When the name of a request parameter from the subsequent authorization request matches the name from the initial PAR request parameter, then the value from the PAR request parameter takes precedence.

For more information about the PAR protocol, see OAuth 2.0 Pushed Authorization Requests on the IETF website.

OAuth Playground

Developers can install and use the PingFederate OAuth Playground to experiment with OAuth and OpenID Connect.

Before you begin

Before you can install the OAuth Playground, install the instance of PingFederate in which you will install the OAuth Playground.

> **Note**
>
> You should not install OAuth Playground in PingFederate production environments.

About this task

Acting as both a client and a resource server, the OAuth Playground simulates authorization requests, responses, and token validation. These features can help you integrate OAuth and OpenID Connect protocols into your applications.

To install the OAuth Playground, perform the following procedure. For more information, including the credentials needed for some OAuth grants, see the `ReadMeFirst` file in the OAuth Playground `.zip` file.

Steps

1. Get the OAuth Playground files:
    1. In a browser, go to the Developer Tools page on the Ping Identity website.
    2. Click the OAuth Playground Download button. If requested, log in to your Ping Identity account. Your browser downloads the OAuth Playground `.zip` file.
    3. Extract the contents from the `.zip` file.
2. Install the OAuth Playground:
    1. Copy the contents of the `/dist/deploy` directory to `/pingfederate/server/default/deploy` in PingFederate.
    2. Merge the contents of the `/dist/conf` directory into `/pingfederate/server/default/conf` in PingFederate.
3. Configure the OAuth Playground:
    1. Open the OAuth Playground by going to https://<pf_host>:9031/OAuthPlayground in a browser.
    2. Click the Setup button. The Setup wizard appears.
    3. Follow the wizard's instructions.

> **Note**
>
> The **Setup** button appears only the first time that you configure the OAuth Playground. To reconfigure the OAuth Playground, click **Settings** on the taskbar. On the **Settings** window, click the **Set Up Again** button. Follow the wizard's instructions.

Web service interfaces and APIs

PingFederate provides two built-in, SOAP-accessible web services related to browser-based single sign-on (SSO).

These services can be used by client applications to manage partner connections and support integration of web applications, respectively.

Connection Management Service

> **Note**
>
> As of PingFederate 10.2, the Connection Management Service has been deprecated and will be removed in a future release.

The Connection Management Service enables creation and deletion of single connection configurations in PingFederate. This service can be used to migrate connections from one server environment to another, for example, from testing or staging to production, or to create new connections in a single server programmatically.

> **Tip**
>
> PingFederate provides a command-line utility that can be used to export and modify connections, as well as other administrative-console configurations, and then import them to target environments. For more information, see Automating configuration migration.

SSO Directory Service

> **Note**
>
> As of PingFederate 10.2, the SSO Directory Service has been deprecated and will be removed in a future release.

The SSO Directory Service provides web application developers with information regarding partner connections and adapter instances.

> **Tip**
>
> Applications accessing the Connection Management Service must first authenticate themselves to the PingFederate server. SSO Directory Service authentication is optional by default, but might be required. For more information, see Configuring service authentication.

Additionally, PingFederate provides REST-based web services and APIs for a variety of administrative and runtime tasks.

OAuth Client Management Service

A runtime API to manage OAuth client applications.

OAuth Access Grant Management Service

A runtime API to retrieve and revoke persistent grants. This API is intended for administrators to manage grants per client or per user.

OAuth Persistence Grant Management API

Another runtime API to retrieve and revoke persistent grants. This API is intended for the use case where clients can assume the responsibility of grant management, provided that the users authorize the clients to do so.

Session Revocation API

A runtime API allowing clients supporting the OpenID Connect protocol to query revocation status of their sessions and add user sessions to the revocation list.

Administrative API

An administrative API to manage various PingFederate settings.

Connection Management Service

The Connection Management Service supports basic connection management capabilities and is accessible only on a PingFederate server running the administrative console.

> **Note**
>
> As of PingFederate 10.2, the Connection Management Service has been deprecated and will be removed in a future release.

The Connection Management Service is useful in a variety of circumstances. Consider the following use cases:

- Using the Connection Management Service as a utility, you can migrate changes to a partner connection through staging environments. For example, development, test, and production.

  Using the Connection Management Service, you might need to make changes to URLs and keys to make the connection appropriate to the next environment.
- Using the Connection Management Service, an external application can update or delete connections programmatically, or create new ones using an exported connection XML file as a template.

You can find the WAR file for this service, `pf-mgmt-ws.war`, in the `<pf_install>/pingfederate/server/default/deploy2` directory.

> **Note**
>
> If you do not want to allow use of the service, do not deploy it: remove the WAR file from the `deploy2` directory.

The SOAP-accessible service endpoint is `pf-mgmt-ws/ws/ConnectionMigrationMgr`.

The web services Description Language (WSDL) document describing this service can be retrieved from `/pf-mgmt-ws/ws/ConnectionMigrationMgr?wsdl`.

Exporting a connection

You can export a connection either manually, using the administrative console, or programmatically through a call to the Connection Management Service.

Whether you export a connection manually or programmatically, the exported XML complies with the standard SAML 2.0 metadata format, with extensions to capture PingFederate's proprietary configuration. Most connection configuration information is contained in the XML markup, with the exception of global configuration items such as adapter instances, datastores, and key pairs. Adapter instances and datastores are referenced by ID, and key pairs are referenced by the MD5 fingerprint of their X.509 certificate. Public certificates, such as the partner's signature verification certificate, are included completely (base-64 encoded).

Export manually

For information about using the administrative console to export connections, see Accessing SP connections or Accessing IdP connections.

Export via the Connection Management Service

The Connection Management Service exposes the following method for exporting connections.

```
public string getConnection( String entityId, String role,) throws IOException
```

The `entityId` parameter is the connection ID, which identifies the connection to be deleted. The `role` parameter is the connection role, the identity provider (IdP) or the service provider (SP).

Code sample

The following example invokes this web service to export a connection.

```
Service service = new Service();
Call call = (Call)service.createCall();
call.setUsername("username");
call.setPassword("password");
call.setTargetEndpointAddress("https://localhost:9999/pf-mgmt-ws/ws/ConnectionMigrationMgr");
call.setOperationName("getConnection");
Object result = call.invoke(new Object[] {"<entityId>", "SP"});
```

Importing connections

Moving a connection from one PingFederate server to another requires care, as the target server must contain the global configuration items, such as datastores, key pairs, and adapter instances, that the connection references.

Changing the references in the XML file, either manually or programmatically, may be necessary to adjust the connection to the target PingFederate environment.

After the required changes are made to the XML file, developers can use the Connection Management Service to import the connection into a different instance of PingFederate.

> **Tip**
>
> Alternatively, you can import XML connection files through the PingFederate administrative console. For more information, see Accessing SP connections or Accessing IdP connections. You can also import the connections into PingFederate manually by copying them into the `<pf_install>/pingfederate/server/default/ data/connection-deployer` directory.
> PingFederate scans this directory periodically and imports connections automatically.

> **Caution**
>
> Manually importing a connection always overwrites an existing connection with the same ID. The web service provides a switch to disallow this behavior, if desired. For more information, see below.

The web service exposes the following method for importing connections.

```
public void saveConnection( String xml, boolean allowUpdate) throws IOException
```

The `xml` parameter is the complete representation of the connection retrieved by your application from an exported connection file, and optionally modified.

If `allowUpdate` is false, the web service can be used only to add a new connection. An error occurs if a connection already exists with the same connection ID and federation protocol in the XML. If `allowUpdate` is true and the connection already exists, it will be overwritten.

Sample code

The following example uses the Apache AXIS libraries to invoke this web service to create a new connection.

```
Service service = new Service();
    Call call = (Call) service.createCall();
    call.setUsername("username");
    call.setPassword("password");
    String addr = "https://localhost:9999/pf-mgmt-ws/ws/ConnectionMigrationMgr";
    call.setTargetEndpointAddress(addr);
    call.setOperationName("saveConnection");
    String xml = "<EntityDescriptor entityID=\"some_entity_id\"
                 ...
                 </EntityDescriptor>";
    boolean allowUpdate = false;
    call.invoke(new Object[]{xml, allowUpdate});
```

Deleting connections

You can invoke the web service to delete connections.

The web service exposes the following method for connection deletion.

```
public void deleteConnection( String entityId, String role) throws IOException
```

The `entityId` parameter is the connection ID, which identifies the connection to be deleted. The `role` parameter is the connection role, identity provider (IdP) or service provider (SP).

Code sample

The following example uses the Apache AXIS libraries to invoke this web service to delete a connection.

```
Service service = new Service();
Call call = (Call) service.createCall();
call.setUsername("username");
call.setPassword("password");
call.setTargetEndpointAddress(
        "https://localhost:9999/pf-mgmt-ws/ws/ConnectionMigrationMgr"
        );
call.setOperationName("deleteConnection");
call.invoke(new Object[]{"entityid", "SP"});
```

Cluster configuration replication

A web service endpoint is available to replicate the administrative-console configuration to other nodes in a PingFederate cluster from the Connection Management Service.

The cluster configuration replication web service endpoint allows a client of this web service to create, update, or delete a connection, and then push the new configuration to the other cluster nodes.

The service endpoint is `/pf-mgmt-ws/ws/ConfigReplication`.

The WSDL document describing this service can be retrieved from `/pf-mgmt-ws/ws/ConfigReplication?wsdl`.

The web service exposes the following method: `public void replicateConfiguration();`.

Code sample

Below is example client code using the Apache AXIS libraries that invokes the configuration replication functionality.

```
Call call2 = (Call) service.createCall();
    call2.setUsername("joe");
    call2.setPassword("test");
    String addr2 = "https://localhost:9999/pf-mgmt-ws/ws/ConfigReplication";
    call2.setTargetEndpointAddress(addr2);
    call2.setOperationName("replicateConfiguration");
    call2.invoke(new Object[]{});
```

Validation disclaimer

Use the administrative console whenever possible to reduce the risk to data integrity.

The import process is not subject to the same rigorous data validation performed by the administrative user interface. Although some checks are made, it is possible to create invalid connections using the connection-migration process. As the XML is complex and validation is limited, use the administrative console to create the initial XML connection instead of attempting to create an XML connection from scratch. That way, changes necessary to the exported connection's XML representation can be held to a minimum, reducing the risk of compromising data integrity.

SSO Directory Service

PingFederate single sign-on (SSO) Directory Service allows applications to retrieve configuration data from a runtime PingFederate server. A PingFederate server in a cluster configured as an administrative console does not support this web service.

> **Note**
>
> As of PingFederate 10.2, the SSO Directory Service has been deprecated and will be removed in a future release.

The SSO Directory Service lets web applications avoid storing and maintaining the data locally.These applications can retrieve the following types of data:

- A list of identity provider (IdP) partners

- A list of service provider (SP) partners
- A list of IdP adapter instances
- A list of SP adapter instances

The SSO Directory Service provides useful information for integrating an application with a PingFederate server. It is a way for the application to dynamically determine which partners can be used for SSO. This means applications do not need to be modified when new partners are configured in PingFederate.

You can find the WAR file for this module, `pf-ws.war`, in the `pingfederate/server/default/deploy` directory.

> **Note**
>
> If you do not want to allow use of the service, remove the WAR file from the `deploy` directory.

The service endpoint is: `pf-ws/services/SSODirectoryService`.

You can retrieve the WSDL document describing this service from `/pf-ws/services/SSODIrectoryService?wsdl`.

You can retrieve a list using any of the following methods:

- `getIDPList` returns a list of active IdP connections configured for SP-initiated SSO. The list contains each IdP's connection ID and connection name.
- `getSPList` returns a list of active SP connections configured for IdP-initiated SSO. The list contains each SP's connection ID and connection name.

  > **Note**
  >
  > For either IdP or SP lists, connection IDs are returned as values for the XML tag `<entityId>`. Connection Names are returned as values for the XML tag <*company*>. For more information see SOAP request and response examples.

- `getAdapterInstanceList` returns a list of SP adapter instances containing an ID and name.
- `getIdpAdapterInstanceList` returns a list of IdP adapter instances containing an ID and name.

  > **Note**
  >
  > These methods do not require input parameters.

The service is also available over HTTP. The query string for retrieving any of the lists is `/pf-ws/services/SSODirectoryService?method=<method_name>`.

Coding example

When you integrate a web application with PingFederate, use the single sign-on (SSO) Directory Service to generate a connection or adapter list. The code needed to create any of the lists is similar.

The following Java code example retrieves an identity provider (IdP) list from the web service. The program calls the getIDPList method in the SSO Directory Service to retrieve an IdP list and print it to the console. This example uses the Apache Axis library and includes optional code for authentication to the PingFederate server. For more information see Configuring service authentication. HTTPS is recommended when including credentials.

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import java.net.URL;
import javax.xml.namespace.QName;
import com.pingidentity.ws.SSOEntity;
public class SSODirectoryClientSample
{
   public static void main(String[] args) throws Exception
   {
    Service service = new Service();
    Call call = (Call) service.createCall();
    call.setUsername("username");
    call.setPassword("pass");
    URL serviceUrl = new URL(
       "https://localhost:9031/pf-ws/services/
         SSODirectoryService");
    QName qn = new QName("urn:BeanService", "SSOEntity");
    call.registerTypeMapping(SSOEntity.class, qn,
      new org.apache.axis.encoding.ser.BeanSerializerFactory(
          SSOEntity.class, qn),

       new org.apache.axis.encoding.ser.BeanDeserializerFactory(
           SSOEntity.class, qn));
    call.setTargetEndpointAddress( serviceUrl );
    call.setOperationName( new QName(
      "http://www.pingidentity.com/servicesSSODirectoryService",
        "getIDPList"));
    Object result = call.invoke( new Object[] {} );
    if (result instanceof SSOEntity[])
    {
       SSOEntity[] idpArray = (SSOEntity[])result;
       for (SSOEntity idp : idpArray)
      {
        System.out.println(idp.getEntityId() + " " +
          idp.getCompany());
      }
    }
    else
    {
       System.out.println("Received problem response from
          server: " + result);
    }
  }
}
```

SOAP request and response examples

A client application must send a SOAP request to the PingFederate server specifying the requested web service and the specific method.

The following is a typical SOAP request for an identity provider (IdP) list using the single sign-on (SSO) Directory Service.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getIDPList
        soapenv:encodingStyle=
            "http://schemas.xmlsoap.org/soap/encoding/"
        xmlns:ns1=
            "https://localhost:9031/ssodir/services/
                 SSODirectoryService"/>
  </soapenv:Body>
</soapenv:Envelope>
```

The PingFederate server's web service returns a response containing the list you requested. The following is an example of a typical SOAP response for an IdP list.

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
  soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getIDPListResponse
        soapenv:encodingStyle=
            "http://schemas.xmlsoap.org/soap/encoding/">
      <getIDPListReturn
          soapenc:arrayType=
             "ns1:IDP[2]" xsi:type="soapenc:Array"
           xmlns:ns1="urn:BeanService"
           xmlns:soapenc=
             "http://schemas.xmlsoap.org/soap/encoding">
         <getIDPListReturn href="#id0" />
         <getIDPListReturn href="#id1" />
      </getIDPListReturn>
    </getIDPListResponse>
    <multiRef id="id0" soapenc:root="0"
        soapenv:encodingStyle=
           "http://schemas.xmlsoap.org/soap/encoding/"
          xsi:type="ns2:IDP"
         xmlns:soapenc=
            "http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:ns2="urn:BeanService">
       <company xsi:type="xsd:string">MegaMarket</company>
      <entityId xsi:type="xsd:string">www.megamarket.com
       </entityId>
    </multiRef>
    <multiRef id="id1" soapenc:root="0"
        soapenv:encodingStyle=
           "http://schemas.xmlsoap.org/soap/encoding/"
          xsi:type="ns3:IDP" xmlns:ns3="urn:BeanService"
         xmlns:soapenc=
            "http://schemas.xmlsoap.org/soap/encoding/">
      <company xsi:type="xsd:string">Ping</company>
      <entityId
          xsi:type="xsd:string">pingfederate3:default:entityId
       </entityId>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

OAuth Client Management Service

PingFederate includes a REST-based web service for OAuth client management.

The OAuth client management service is provided primarily for organizations with several OAuth clients to allow programmatic management of OAuth clients and as an alternative to using the administrative console, the administrative API, or dynamic client registration.

The Endpoint: `/pf-ws/rest/oauth/clients` and Endpoint: `/pf-ws/rest/oauth/clients/<clientId>` REST resources are URL path extensions of the PingFederate runtime endpoint:

- https://www.example.com:9031/pf-ws/rest/oauth/clients

- https://www.example.com:9031/pf-ws/rest/oauth/clients/<*clientId*>

Important

The OAuth Client Management Service requires use of external storage for client records.

Note

Applications must authenticate to this web service using HTTP Basic authentication and credentials that are validated through a password credential validator (PCV) instance. The PCV instance, in turn, must be selected in the OAuth authorization server configuration.

Tip

The administrative API can also manage OAuth clients programmatically regardless of whether the client records are managed in XML files or in a database.

Endpoint: `/pf-ws/rest/oauth/clients`

This endpoint accepts the POST, PUT, and GET methods. The POST and PUT methods described in this section require parameter name-value pairs formatted in JSON.

POST

Use the POST method to create a new client based on the parameters provided in the request. Parameters correspond to the fields on the Client page. The required `MIME` type is `application/json`.

JSON Parameters

Parameter	Description
`clientId` (Required)	A unique identifier the client provides to the resource server to identify itself. This identifier is included with every request the client makes.
`enabled`	Specifies whether the client is enabled. Valid values are `true` or `false`. The default value is `true`.
`name` (Required)	A descriptive name for the client instance. This name appears when the user is prompted for authorization.
`description`	A description of what the client application does. This description appears when the user is prompted for authorization.

Parameter	Description
`clientAuthnType`	The authentication method that the client uses. • Set to `none` if your use case doesn't require client authentication. ⓘ **Note** A value other than `none` is required for any of the following use cases: ◦ This client uses the `client_credentials` grant type. Refer to the `grantTypes` parameter. ◦ This client signs its ID tokens using an HMAC signing algorithm. Refer to the `idTokenSigningAlgorithm` parameter. ◦ This client can access the Session Revocation API endpoint. Refer to the `grantAccessSessionRevocationApi` parameter. ◦ This client sends a `secret` parameter value. • Set to `SECRET` for HTTP Basic authentication. This authentication method requires the `secret` or `client_secret_jwt` parameter. • Set to `CLIENT_CERT` for mutual SSL/TLS authentication. This value is recommended for client applications where security policies prohibit storing passwords. This authentication method requires the `clientCertIssuerDn` and `clientCertSubjectDn` parameters. **Important** If you select mutual SSL/TLS authentication, you must configure a secondary PingFederate HTTPS port. You can find more information about the `pf.secondary.https.port` property in the description in the table in Configuring PingFederate properties. • Set to `PRIVATE_KEY_JWT` if the client authenticates through the private_key_jwt client authentication method as defined in Client Authentication in the OpenID Connect (OIDC) specification • Set to `CLIENT_SECRET_JWT` if the client authenticates through the client_secret_jwt client authentication method as defined in Client Authentication in the OIDC specification.
`secret`	The client password or phrase. Required when the `clientAuthnType` parameter is set to `SECRET` or `CLIENT_SECRET_JWT`.
`clientCertIssuerDn`	The issuer distinguished name (DN) of the client certificate. These are certificate authority (CA) certificates imported into PingFederate on the Security > Certificate & Key Management > Trusted CAs page. Alternatively, it might be set to `Trust Any` to trust all of the issuers found on the Trusted CAs page. Required when the `clientAuthnType` parameter is set to `CLIENT_CERT`.
`clientCertSubjectDn`	The subject DN of the client certificate. Required when the `clientAuthnType` parameter is set to `CLIENT_CERT`.

Parameter	Description
`tokenEndpointAuthSigningAlgorithm`	The signing algorithm that the client must use to sign the JSON Web Token (JWT) for client authentication. Applicable only when the `clientAuthnType` parameter is provided with a value of `PRIVATE_KEY_JWT` or `CLIENT_SECRET_JWT`. Allowed values: • `RS256` : RSA using SHA-256 • `RS384` : RSA using SHA-384 • `RS512` : RSA using SHA-512 • `HS256` : HMAC using SHA-256 • `HS384` : HMAC using SHA-384 • `HS512` : HMAC using SHA-512 • `ES256` : ECDSA using P256 Curve and SHA-256 • `ES384` : ECDSA using P384 Curve and SHA-384 • `ES512` : ECDSA using P512 Curve and SHA-512 • `PS256` : RSASSA-PSS using SHA-256 • `PS384` : RSASSA-PSS using SHA-384 • `PS512` : RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment or an integration with an Hardware Security Modules (HSM) and a static-key configuration for OAuth and OIDC. You can find information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. If this parameter isn't provided, the client can use any of the supported signing algorithms.
`enforceReplayPrevention`	Determines whether PingFederate mandates a unique signed JWT from the client for each request when the client is configured to authenticate using the private_key_jwt client authentication method to transmit request parameters using signed request objects or to do both. Valid values are `true` or `false`. ⓘ **Note** The underlying Assertion Replay Prevention Service is cluster-aware. Learn more in Assertion Replay Prevention Service.

Parameter	Description
`requireSignedRequests`	**Determines whether the client must transmit request parameters in a single, self-contained parameter.** **The parameter name is `request`.** **The value of the `request` parameter is a signed JWT whose claims represent the request parameters of the authorization request. The OIDC specification calls this JWT a request object.** **Valid values are `true` or `false`.** **Note** If a client includes in an authorization request a request parameter other than `client_id` as a parameter outside of the signed request object and a claim inside of the signed request object, PingFederate always uses the claim value found inside the signed request object to process the request further. For the `client_id` request parameter, the values outside of the signed request object must match the claim values inside of the signed request object. If the values don't match, PingFederate returns an error message to the client. If a request parameter is found only outside of the signed request object, PingFederate ignores the request parameter and returns no error message. **Tip** Per OAuth and OIDC specifications, a client must always include in an authorization request the `client_id` parameter outside of the signed request object. **Learn more about request objects in RFC 9101: JWT Secured Authorization Request (JAR).** **If this parameter isn't provided, the default value of `false` applies.**

Parameter	Description
`requestObjectSigningAlgorithm`	The signing algorithm that the client must use to sign its request objects for transmission of request parameters. Applicable only when the client might send its authorization requests using request objects. Allowed values: • `RS256` : RSA using SHA-256 • `RS384` : RSA using SHA-384 • `RS512` : RSA using SHA-512 • `HS256` : HMAC using SHA-256 • `HS384` : HMAC using SHA-384 • `HS512` : HMAC using SHA-512 • `ES256` : ECDSA using P256 Curve and SHA-256 • `ES384` : ECDSA using P384 Curve and SHA-384 • `ES512` : ECDSA using P521 Curve and SHA-512 • `PS256` : RSASSA-PSS using SHA-256 • `PS384` : RSASSA-PSS using SHA-384 • `PS512` : RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment or an integration with an HSM and a static-key configuration for OAuth and OIDC. You can find information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. If this parameter isn't provided, the client can use any of the supported signing algorithms.
`jwksUrl` or `jwks`	The URL of the JSON Web Key Set (JWKS) or the actual JWKS from the client. • If the client is configured to use the `private_key_jwt` or `client_secret_jwt` client authentication method to transmit request parameters in signed request objects or to transmit CIBA request parameters in signed request objects, only one of the previous values is required for PingFederate to verify the authenticity of the JWTs. Either value can be defined even if the client isn't configured to use JWTs for authentication or transmission of request parameters. This flexibility allows the client to transmit request parameters in signed request objects for some requests and without the use of signed request objects for some other transactions. • If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 8 or Java 11 runtime environment or integrated with an HSM and a static-key configuration for OAuth and OIDC. Learn more about runtime processing in Authorization endpoint. • If the client is configured to encrypt ID tokens using an asymmetric encryption algorithm, either the JWKS URL or the actual JWKS must be provided. Refer to the ID Token Key Management Encryption Algorithm setting.

Parameter	Description
`redirectUris`	URIs where the OAuth AS may redirect the resource owner's user agent after authorization is obtained. The authorization code and implicit grant types require at least one redirection URI.
`logoUrl`	The location of the logo used on user-facing OAuth grant authorization and revocation pages. For best results with the installed HTML templates, the recommended size is 72 x 72 pixels.
`bypassApprovalPage`	If set to `true`, resource-owner approval for client access is assumed, and PingFederate no longer presents to the user an authorization consent page or redirects to a trusted web application that is responsible to prompt the user for authorization for this client. A valid value are `true` or `false`. If this parameter isn't provided, the default value of `false` applies.
`restrictScopes`	Controls whether all existing common scopes and scope groups and those created in the future or only the selected ones should be made available to the client. Valid value are `true` or `false`. When set to `true`, PingFederate limits the client to a list of common scopes and scope groups as specified by the `restrictedScopes` parameter. When set to `false`, all existing common scopes and scope groups and those created in the future are available to the client. If this parameter isn't provided, the default value of `false` applies. ⓘ **Note** Depending on the configured dynamic scope patterns and whether they are defined as common or exclusive dynamic scopes, this setting and the `restrictedScopes` parameter value could impact the results of scope evaluation. The default scope, however, is always allowed for and available to all clients. You can find detailed information in the Dynamic scope evaluation and per-client scope management section in Scopes and scope management.
`restrictedScopes`	Used in conjunction with the `restrictScopes` parameter value of `true` to limit this client to a list of common scopes or scope groups in addition to the default scope. Scopes and scope groups that aren't listed or are created in the future become invalid for the client. If the client tries to use an excluded scope or scope group, it receives an `invalid_scope` error message from PingFederate.

Parameter	Description
`exclusiveScopes`	This setting controls whether any exclusive scopes and scope groups should be made available to the client. As needed, provide this parameter with a list of exclusive scopes or scope groups that are intended for the client. Excluded scopes and scope groups and those created in the future become invalid for the client. If the client tries to use an excluded scope or scope group, it will receive an `invalid_scope` error message from PingFederate. If this parameter isn't provided, no exclusive scopes or scope groups are available to the client. **Note** Depending on the configured dynamic scope patterns and whether they are defined as common or exclusive dynamic scopes, this setting can impact the results of scope evaluation. The default scope is always allowed for and available to all clients. You can find detailed information in the **Dynamic scope evaluation and per-client scope management** section in Scopes and scope management.
`grantTypes`	An array of one or more grant types, which a client can request. PingFederate accepts the following values: • `authorization_code` • `implicit` • `refresh_token` • `client_credentials` • `urn:ietf:params:oauth:grant-type:device_code` • `urn:openid:params:grant-type:ciba` • `password` • `extension` (JWT Bearer Token or SAML 2.0 Bearer Assertion) Learn more about each grant type in Grant types.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>restrictedResponseTypes</td><td>An array of one or more response types, which a client can request.
PingFederate accepts the following values:
<ul>
<li>code</li>
<li>code id_token</li>
<li>code id_token token</li>
<li>code token</li>
<li>id_token</li>
<li>id_token token</li>
<li>token</li>
</ul>
Learn more about these response types in Definitions of Multiple-Valued Response Type Combinations.
If one or more response types are specified, the resulting client is only allowed to send one of the specified response types at runtime. Requests from this client with other response types will be rejected.
Response type and grant type parameters must be provided in tandem because certain response types require one or more grant types (and inversely). The following table provides a summary of their relationship:
<table>
<tr><th>response type</th><th>grant types</th></tr>
<tr><td>code</td><td>authorization_code</td></tr>
<tr><td>code id_token</td><td>authorization_code and implicit</td></tr>
<tr><td>code id_token token</td><td>authorization_code and implicit</td></tr>
<tr><td>code token</td><td>authorization_code and implicit</td></tr>
<tr><td>id_token</td><td>implicit</td></tr>
<tr><td>id_token token</td><td>implicit</td></tr>
<tr><td>token</td><td>implicit</td></tr>
</table>
</td></tr>
<tr><td>defaultAccessTokenManagerId</td><td>Determines the default Access Token Management (ATM) instance for this client.</td></tr>
<tr><td>validateUsingAllEligibleAtms</td><td>Applicable only to resource server clients.
If selected, this resource server client isn't required to specify the additional access_token_manager_id, aud, or resource parameters to disambiguate the ATM instance in its token validation requests. When the resource server client doesn't specify the desired ATM instance, PingFederate validates the access tokens against all eligible ATM instances. This simplifies interactions with PingAccess by avoiding the need to align resource URIs between PingAccess and PingFederate.
This checkbox is cleared by default.</td></tr>
</table>

Parameter	Description
`requireProofKeyForCodeExchange`	Applicable when the client is configured to support the `authorization_code` grant type. Valid values are `true` or `false`. Determines whether the client must provide certain parameters to reduce the risk of authorization code interception attack. Learn more in the Proof Key for Code Exchange (PKCE) by OAuth Public Clients specification. When enabled, this client must include a one-time string value using the `code_challenge` parameter in its authorization request. Learn more in Authorization endpoint. It must also submit the corresponding code verifier through the `code_verifier` parameter in its token request when exchanging an authorization code for an access token. Learn more in OAuth grant type parameters. If this parameter isn't provided, the default value of `false` applies.
`persistentGrantExpirationType`	Overrides the Persistent Grant Max Lifetime value set globally in System > OAuth Settings > Authorization Server Settings. **Note** This setting can be overridden per grant-mapping configuration using an extended persistent grant attribute `PERSISTENT_GRANT_LIFETIME`. The `PERSISTENT_GRANT_LIFETIME` attribute is defined in **System > OAuth Settings > Authorization Server Settings**. When this attribute is active, you can set the lifetime of persistent grants based on the outcome of attribute mapping expressions in individual grant-mapping configurations. For grant-mapping configurations that don't require this fine-grain control, you can configure them to use the default value.
`persistentGrantExpirationTime`	An integer representing units of time for storage of persistent grants for this client. Required when the `persistentGrantExpirationType` parameter is provided with a value of `OVERRIDE_SERVER_DEFAULT`.
`persistentGrantExpirationTimeUnit`	Units for the expiration time set by the `persistentGrantExpirationTime` parameter. Allowed values: • `h` (hours) • `d` (days) • `n` (minutes) Required when the `persistentGrantExpirationType` parameter is provided with a value of `OVERRIDE_SERVER_DEFAULT`.

Parameter	Description
`persistentGrantIdleTimeoutType`	Overrides the Persistent Grant Idle Timeout field value set globally in System > OAuth Settings > Authorization Server Settings. Allowed values: • `SERVER_DEFAULT` (default value): Use the global setting. • `NONE` : Grants do not expire due to inactivity. • `OVERRIDE_SERVER_DEFAULT` : Use in conjunction with the `persistentGrantIdleTimeout` and `persistentGrantIdleTimeoutUnit` parameters to set the idle timeout window. If an idle timeout value is configured, the idle timeout window slides when a persistent grant is updated. Learn more in Transient grants and persistent grants. If you configure an idle timeout value, the idle timeout window slides when a persistent grant updates. When you have an idle timeout value configured without a maximum lifetime, persistent grants remain valid until they expire because of inactivity or until the grant storage revokes or removes them. When you have an idle timeout value configured with a maximum lifetime, persistent grants remain valid until they expire due to inactivity or lifetime expiration or until the grant storage removes them.
`persistentGrantIdleTimeout`	An integer representing the inactivity timeout value for this client. Required when the `persistentGrantIdleTimeoutType` parameter is provided with a value of `OVERRIDE_SERVER_DEFAULT` .
`persistentGrantIdleTimeoutTimeUnit`	Units for the inactivity timeout value set by the `persistentGrantIdleTimeout` parameter. Allowed values: • `h` (hours) • `d` (days) • `n` (minutes) Required when the `persistentGrantIdleTimeoutType` parameter is provided with a value of `OVERRIDE_SERVER_DEFAULT` .
`refreshRolling`	Overrides the Roll Refresh Token Values setting configured globally in System > OAuth Settings > Authorization Server Settings. Valid values are `true` or `false` . ⓘ **Note** A value of `true` doesn't override the **Minimum Interval to Roll Refresh Tokens** value set on the **Authorization Server Settings** page. If this parameter isn't provided, the Roll Refresh Token Values setting configured globally in the System > OAuth Settings > Authorization Server Settings page is used.
`refreshTokenRollingIntervalType`	When set to the default value, `SERVER_DEFAULT` , the client's minimum refresh token rolling interval comes from the global value in the Minimum Interval to Roll Refresh Tokens field on the Authorization Server Settings page. When set to `OVERRIDE_SERVER_DEFAULT` , the client's refresh token rolling interval comes from the value of the `refreshTokenRollingInterval` parameter.

Parameter	Description
`refreshTokenRollingInterval`	The minimum number of hours that must pass before a new refresh token can be issued. This value overrides the global value in the Minimum Interval to Roll Refresh Tokens field on the Authorization Server Settings page when the `refreshTokenRollingIntervalType` parameter is set to `OVERRIDE_SERVER_DEFAULT`. Valid values are an integer between `0` - `8760`. Required when the `refreshTokenRollingInterval` parameter is set to `OVERRIDE_SERVER_DEFAULT`.
`refreshTokenRollingIntervalTimeUnit`	Units for the refresh token rolling interval set by the `refreshTokenRollingInterval` parameter. Valid values: • `d` (days) • `h` (hours) • `m` (minutes) The default value is `h`. An integer between `0` to `8760` that represents the minimum number of hours that must pass before a new refresh token can be issued. This value overrides the global value in the Minimum Interval to Roll Refresh Tokens field on the Authorization Server Settings page when the `refreshTokenRollingIntervalType` parameter is set to `OVERRIDE_SERVER_DEFAULT`. Required when the `refreshTokenRollingInterval` parameter is set to `OVERRIDE_SERVER_DEFAULT`.
`refreshTokenRollingGracePeriod`	The amount of time in seconds that a rolled refresh token is still valid in the event that the client failed to receive an updated one during a roll. **Note** When a new refresh token is issued, it is the only valid token and invalidates the previous token.
`requirePushedAuthorizationRequests`	When set to `true` the client must use the pushed authorization requests (PAR) endpoint `/as/par.oauth2` on the AS to initiate authorization flows. When set to `false`, the client can use the PAR endpoint. The default value is `false`. This parameter works in conjunction with the PAR Status setting on the AS. For example: • If PAR is Enabled on the AS and this parameter is set to `true`, the client must use PAR • If PAR is Enabled on the AS but this parameter is set to `false`, the client can use PAR • If PAR is Required on the AS but this parameter is set to `false`, the client must use PAR • If PAR is Disabled on the AS and this parameter is set to `true`, the client can't access the AS Don't set this parameter to `true` if PAR is disabled on the AS. Learn more about PAR in Pushed authorization requests endpoint and Configuring authorization server settings.

Parameter	Description
`requireDpop`	Specifies whether the client must use the OAuth 2.0 Demonstrating Proof of Possession (DPoP) protocol for authentication. Valid values are `true` or `false`. The default value is `false`. The protocol is specified in OAuth 2.0 Demonstrating Proof of Possesion (DPoP) The Authorization Server Settings page includes settings for determining DPoP behavior. Learn more in Configuring authorization server settings.
OIDC client settings **Note** The following parameters are only applicable when this client supports the OIDC use cases.	
`idTokenSigningAlgorithm`	The JSON Web Signature (JWS) algorithm required for the OIDC tokens. Allowed values: • `none`: No signing algorithm • `HS256`: HMAC using SHA-256 • `HS384`: HMAC using SHA-384 • `HS512`: HMAC using SHA-512 • `ES256`: ECDSA using P256 Curve and SHA-256 • `ES384`: ECDSA using P384 Curve and SHA-384 • `ES512`: ECDSA using P521 Curve and SHA-512 • `RS256`: RSA using SHA-256 • `RS384`: RSA using SHA-384 • `RS512`: RSA using SHA-512 • `PS256`: RSASSA-PSS using SHA-256 • `PS384`: RSASSA-PSS using SHA-384 • `PS512`: RSASSA-PSS using SHA-512 **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment or an integration with an HSM and a static-key configuration for OAuth and OIDC. You can find information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. **Important** If static keys for OAuth and OIDC are enabled, use either an RSA algorithm or an EC algorithm that has been configured with an active static key.

Parameter	Description
`idTokenEncryptionAlgorithm`	The algorithm used to encrypt or otherwise determine the value of the content encryption key. Allowed values: • `dir` : Direct Encryption with symmetric key • `A128KW` : AES-128 Key Wrap • `A192KW` : AES-192 Key Wrap • `A256KW` : AES-256 Key Wrap • `A128GCMKW` : AES-GCM-128 key encryption • `A192GCMKW` : AES-GCM-192 key encryption • `A256GCMKW` : AES-GCM-256 key encryption • `ECDH-ES` : ECDH-ES • `ECDH-ES+A128KW` : ECDH-ES with AES-128 Key Wrap • `ECDH-ES+A192KW` : ECDH-ES with AES-192 Key Wrap • `ECDH-ES+A256KW` : ECDH-ES with AES-256 Key Wrap • `RSA-OAEP` : RSAES-OAEP • `RSA-OAEP-256` : RSAES OAEP using SHA-256 and MGF1 with SHA-256
`idTokenContentEncryptionAlgorithm`	The content encryption algorithm used to perform authenticated encryption on the plain text payload of the token. Required if an algorithm is provided through the `idTokenEncryptionAlgorithm` parameter. Allowed values: • `A128CBC-HS256` : Composite AES-CBC-128 HMAC-SHA-256 • `A192CBC-HS384` : Composite AES-CBC-192 HMAC-SHA-384 • `A256CBC-HS512` : Composite AES-CBC-256 HMAC-SHA-512 • `A128GCM` : AES-GCM-128 • `A192GCM` : AES-GCM-192 • `A256GCM` : AES-GCM-256
`policyGroupId`	The desired OIDC policy.
`grantAccessSessionRevocationApi`	Set to `true` to allow this client to access the Session Revocation API for back-channel session query and revocation. Valid values are `true` or `false` . If this parameter isn't provided, the default value of `false` applies. **Note** If clients are allowed to add sessions to the revocation list, you can enable the **Check session revocation status** option in the applicable Access Token Management instances for the token validation process to consider whether a session has been added to the revocation list.
`pairwiseUserType`	Set to `true` to allow this client to use pairwise pseudonymous IDs. This parameter is set to `false` by default.
`sectorIdentifierUri`	Specify one HTTPS URL only. This parameter is applicable only if `pairwiseUserType` is set to `true` .

Parameter	Description
Note If the **Track User Sessions for Logout** checkbox is selected in the **System > OAuth Settings > Authorization Server Settings** page, you can provide the following two additional parameters to enable asynchronous front-channel logout for this client.	
`pingAccessLogoutCapable`	If set to `true`, PingFederate sends logout requests through the browser to an OIDC endpoint in PingAccess as part of the logout process. Valid values are `true` or `false`. If this parameter isn't provided, the default value of `false` applies.
`logoutUris`	A list of additional endpoints at the relying parties as needed. When the Logout Mode is set to OIDC Front-Channel or Ping Front-Channel, PingFederate sends requests to these URIs through the browser as part of the logout process. For Ping Front-Channel mode, the relying parties must return an image in their logout responses. Otherwise, PingFederate returns an error message or redirects to the `InErrorResource` parameter value, if specified.
`postLogoutRedirectUris`	URIs to which PingFederate can redirect the user after a logout is performed if the relying party's logout request includes the `post_logout_redirect_uri` parameter, and it matches one of the URIs configured in this field.
Device Authorization Grant client settings	

Parameter	Description
`deviceFlowSettingType`	Controls whether to use global device authorization grant settings defined on the System > OAuth Settings >Authorization Server Settings page. Valid values are `SERVER_DEFAULT` and `OVERRIDE_SERVER_DEFAULT`. Set to `OVERRIDE_SERVER_DEFAULT` and configure any of the following settings: *userAuthzUrlOverride* This field controls whether PingFederate should use a different URL, such as for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses. Learn more in Device authorization endpoint. For example, if this field is configured with a value of https://www.example.org/welcome, PingFederate returns https://www.example.org/welcome and https://www.example.org/welcome?user_code=<*activationcode*> as the verification URIs. After processing the device authorization response, which includes the verification URIs, the device presents one of them to the user. The user is expected to browse to the presented verification URI on a second device. **Important** The target web server must redirect the browser to PingFederate at its user authorization endpoint. Learn more in User authorization endpoint. It must also preserve the `user_code` parameter value, if provided. For example, if the base URL of your PingFederate server is https://www.example.com, and this field is configured with a value of https://www.example.org/welcome, the target web server must redirect as follows: • https://www.example.org/welcome to https://www.example.com/as/user_authz.oauth2 • https://www.example.org/welcome?user_code=<*activationcode*> to https://www.example.com/as/user_authz.oauth2?user_code=<*activationcode*> *pendingAuthzTimeoutOverride* The lifetime of an activation code (the `user_code` parameter value) in seconds. *devicePollingIntervalOverride* The amount of time in seconds that the device waits between polling requests to the PingFederate token endpoint. *bypassActivationCodeConfirmationOverride* When PingFederate receives a verification request that includes an activation code (the `user_code` parameter value), it prompts the user to confirm the activation code. This field controls whether PingFederate should skip this confirmation step. Set to `true` if you want PingFederate to skip the confirmation step.
Client-initiated backchannel authentication (CIBA) client settings	

Parameter	Description
`cibaTokenDeliveryMode`	The token delivery method that the client supports. PingFederate supports `poll` and `ping`. Set to `poll` if the client can check for the authorization results periodically at the token endpoint. Set to `ping` if the client prefers to wait for a ping callback message from PingFederate as a signal that the authorization result is ready for pickup. If the client-initiated backchannel authentication (CIBA) grant type is enabled, this parameter is required. No value is applied.
`cibaNotificationEndpoint`	The client's notification endpoint to which PingFederate sends its ping call-back messages. Required only if `ping` is the configured token delivery method.
`cibaPollingInterval`	Specifies the number of seconds that the client must wait between its attempts to check for the authorization results at the token endpoint. When PingFederate receives a token request within this time interval, it returns a `slow_down` error message to the client. Valid values are an integer from `1` - `3600`. If the CIBA grant type is enabled, this parameter is required. No value is applied.
`cibaPolicyId`	Specifies the CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication request, such as the maximum lifetime of authentication requests, the validity of unsigned login hint tokens, and the mapping configuration of identity hints. Provides an existing CIBA policy. If this parameter isn't provided and the CIBA grant type is enabled, PingFederate associates the client with the CIBA request policy that has been designated as the default CIBA request policy on the Applications > OAuth > CIBA Request Policies page.
`cibaUserCodeSupported`	Indicates whether the client supports user code. The purpose of this code is to authorize the transmission of an authentication request to the user's authentication device. Valid values are `true` or `false`. If this parameter isn't provided and the CIBA grant type is enabled, user code support is not enabled. When user code support is enabled, the associated CIBA request policy must also be user code enabled.

Parameter	Description
`cibaRequireSignedRequests`	Determines whether the client must transmit request parameters in a single, self-contained parameter. The parameter name is `request`. The value of the `request` parameter is a signed JWT whose claims represent the request parameters of the authorization request. The OIDC specification calls this JWT a request object. Valid values are `true` or `false`. If this parameter isn't provided and the CIBA grant type is enabled, CIBA signed requests are not required. If CIBA signed requests are required, the client must also be configured with either the JWKS URL or the actual JWKS from the client.
`cibaRequestObjectSigningAlgorithm`	The signing algorithm that the client must use to sign its request objects for transmission of request parameters. Allowed values: • `RS256`: RSA using SHA-256 • `RS384`: RSA using SHA-384 • `RS512`: RSA using SHA-512 • `HS256`: HMAC using SHA-256 • `HS384`: HMAC using SHA-384 • `HS512`: HMAC using SHA-512 • `ES256`: ECDSA using P256 Curve and SHA-256 • `ES384`: ECDSA using P384 Curve and SHA-384 • `ES512`: ECDSA using P521 Curve and SHA-512 • `PS256`: RSASSA-PSS using SHA-256 • `PS384`: RSASSA-PSS using SHA-384 • `PS512`: RSASSA-PSS using SHA-512 ⓘ **Note** RSASSA-PSS signing algorithms require a Java 8 or Java 11 runtime environment or an integration with an HSM and a static-key configuration for OAuth and OIDC. You can find information on HSM integration and static keys in Supported hardware security modules and Keys for OAuth and OpenID Connect, respectively. If this parameter isn't provided and the CIBA grant type is enabled, the client can use any of the allowed signing algorithms.
Token introspection settings	

Parameter	Description
`introspectionSigningAlgorithm`	The JWS algorithm used to sign token introspection responses. This parameter is optional. Allowed values: • `none` : No signing algorithm • `HS256` : HMAC using SHA-256 • `HS384` : HMAC using SHA-384 • `HS512` : HMAC using SHA-512 • `ES256` : ECDSA using P256 Curve and SHA-256 • `ES384` : ECDSA using P384 Curve and SHA-384 • `ES512` : ECDSA using P521 Curve and SHA-512 • `RS256` : RSA using SHA-256 • `RS384` : RSA using SHA-384 • `RS512` : RSA using SHA-512 • `PS256` : RSASSA-PSS using SHA-256 • `PS384` : RSASSA-PSS using SHA-384 • `PS512` : RSASSA-PSS using SHA-512 The default value is `RS256` .
`introspectionEncryptionAlgorithm`	The JSON Web Encryption (JWE) algorithm used to encrypt the content-encryption key of token introspection responses. This parameter is optional. Allowed values: • `dir` : Direct Encryption with symmetric key • `A128KW` : AES-128 Key Wrap • `A192KW` : AES-192 Key Wrap • `A256KW` : AES-256 Key Wrap • `A128GCMKW` : AES-GCM-128 key encryption • `A192GCMKW` : AES-GCM-192 key encryption • `A256GCMKW` : AES-GCM-256 key encryption • `ECDH-ES` : ECDH-ES • `ECDH-ES+A128KW` : ECDH-ES with AES-128 Key Wrap • `ECDH-ES+A192KW` : ECDH-ES with AES-192 Key Wrap • `ECDH-ES+A256KW` : ECDH-ES with AES-256 Key Wrap • `RSA-OAEP` : RSAES-OAEP • `RSA-OAEP-256` : RSAES OAEP using SHA-256 and MGF1 with SHA-256 For asymmetric algorithms, a JWKS or the HTTPS URL of a JWKS endpoint is required. For symmetric algorithms, the reversible secret is required.

Parameter	Description
`introspectionContentEncryptionAlgorithm`	The JWE content-encryption algorithm for token introspection responses. This parameter's value must be set if the `introspectionEncryptionAlgorithm` parameter is set. Allowed values: • `A128CBC-HS256` : Composite AES-CBC-128 HMAC-SHA-256 • `A192CBC-HS384` : Composite AES-CBC-192 HMAC-SHA-384 • `A256CBC-HS512` : Composite AES-CBC-256 HMAC-SHA-512 • `AES_128_GCM` • `AES_192_GCM` • `AES_256_GCM`
JWT secured authorization response mode (JARM) settings	
`requireJwtSecuredAuthorizationResponseMode`	When enabled, the client must use JARM. The client's authorization requests must include one of the following authorization response mode values: • `jwt` : ◦ Query JWT response in the case of the authorization code grant ◦ Fragment JWT response in the case of the implicit grant • `query.jwt` : ◦ JWT response in the case of the authorization code grant ◦ Failure in the case of the implicit grant unless the response is an encrypted JWT based on the client settings • `fragment.jwt` : ◦ Fragment JWT response • `form_post.jwt` : ◦ Auto form-post JWT response JARM is a mechanism to enhance the security of the standard authorization response. It adds support for signing and encryption, sender authentication, and audience restriction. It also offers protection from replay, credential leakage, and mix-up attacks. JARM can be combined with any response type. Learn more in the JARM specification. Valid values are `true` or `false` . The default value is `false` .

Parameter	Description
`authorizationResponseSigningAlgorithm`	The JWS algorithm that PingFederate uses to sign JARM authorization responses. This parameter is optional. Allowed values: • `none` : No signing algorithm • `HS256` : HMAC using SHA-256 • `HS384` : HMAC using SHA-384 • `HS512` : HMAC using SHA-512 • `ES256` : ECDSA using P256 Curve and SHA-256 • `ES384` : ECDSA using P384 Curve and SHA-384 • `ES512` : ECDSA using P521 Curve and SHA-512 • `RS256` : RSA using SHA-256 • `RS384` : RSA using SHA-384 • `RS512` : RSA using SHA-512 • `PS256` : RSASSA-PSS using SHA-256 • `PS384` : RSASSA-PSS using SHA-384 • `PS512` : RSASSA-PSS using SHA-512 The default value is `RS256` .
`authorizationResponseEncryptionAlgorithm`	The JWE encryption algorithm used to encrypt the content-encryption key of JARM authorization responses. This parameter is optional. Allowed values: • `dir` : Direct Encryption with symmetric key • `A128KW` : AES-128 Key Wrap • `A192KW` : AES-192 Key Wrap • `A256KW` : AES-256 Key Wrap • `A128GCMKW` : AES-GCM-128 key encryption • `A192GCMKW` : AES-GCM-192 key encryption • `A256GCMKW` : AES-GCM-256 key encryption • `ECDH-ES` : ECDH-ES • `ECDH-ES+A128KW` : ECDH-ES with AES-128 Key Wrap • `ECDH-ES+A192KW` : ECDH-ES with AES-192 Key Wrap • `ECDH-ES+A256KW` : ECDH-ES with AES-256 Key Wrap • `RSA-OAEP` : RSAES-OAEP • `RSA-OAEP-256` : RSAES OAEP using SHA-256 and MGF1 with SHA-256 For asymmetric algorithms, a JWKS or the HTTPS URL of a JWKS endpoint is required. For symmetric algorithms, the reversible secret is required.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>authorizationResponseContentEncryptionAlgorithm</td><td>The JWE content-encryption algorithm for JARM authorization responses.
This parameter's value must be set if the authorizationResponseEncryptionAlgorithm parameter is set.
This parameter's value must be null if the authorizationResponseEncryptionAlgorithm parameter is not set.
Allowed values:
• AES_128_CBC_HMAC_SHA_256
• AES_192_CBC_HMAC_SHA_384
• AES_256_CBC_HMAC_SHA_512
• AES_128_GCM
• AES_192_GCM
• AES_256_GCM</td></tr>
<tr><td colspan="2"><b>Extended properties</b></td></tr>
<tr><td>extendedParameters</td><td>Provides values for extended client metadata fields.

```
{
  ...
  "extendedParams": {
    "entry": [
      {
        "key": "ContactName",
        "value": {
          "elements": "J. Smith"
        }
      },
      {
        "key": "ContactNumbers",
        "value": {
          "elements": [
            "555-123-4567",
            "555-987-6543"
          ]
        }
      }
    ]
  },
  ...
}
```

This sample request provides a value for a single-valued client metadata field, ContactName, and multiple values for a multi-valued client metadata field, ContactNumbers.
Extended client metadata fields are defined on the System > Server Extended Properties page.</td></tr>
</table>

Sample JSON

```
{
  "client": [
    {
      "secret": "L1u508MfeZYTvR03kcpa6ezysNEspFEtzxSAIEOTll8AuNd2pnNqjkRdOXzfTFXc",
      "clientId": "SampleClient",
      "description": "This is a sample client.",
      "grantTypes": [
        "refresh_token",
        "authorization_code"
      ],
      "name": "Sample Client",
      "redirectUris": [
        "https://www.example.com/redirect1",
        "https://www.example.com/redirect2"
      ]
    }
  ]
}
```

Return codes

- `200 – Success`
- `400 – Failed To Create Client`

  The response contains details as to why the client creation failed.
- `401 – Invalid Credentials`

  The user doesn't exist or isn't authorized to create a client.
- `500 – Internal Server Error`

  An unknown error has occurred.

PUT

Updates client details for a specified client.

> **Note**
>
> You can't update a client ID value. You can delete the client record and create a new one with a new client ID value.

JSON Parameters

The same parameters described for POST apply for PUT with one addition: `forceSecretChange`.

Use this parameter, set to `true`, in conjunction with the `secret` parameter to change a client pass phrase.

Valid values are `true` or `false`. If this parameter isn't provided, the default value of `false` applies.

Note

If the `secret` parameter is used without a `forceSecretChange` parameter value of `true`, the `secret` value is ignored.

Sample JSON

```
{
  "client": [
    {
      "secret": "L1u508MfeZYTvR03kcpa6ezysNEspFEtzxSAIEOTll8AuNd2pnNqjkRdOXzfTFXc",
      "forceSecretChange": "true",
      "clientId": "SampleClient",
      "description": "This is a sample client.",
      "grantTypes": [
        "refresh_token",
        "authorization_code"
      ],
      "name": "Sample Client",
      "redirectUris": [
        "https://www.example.com/redirectOne",
        "https://www.example.com/redirectTwo"
      ]
    }
  ]
}
```

Return codes

- `200 – Success`

  The body contains a list of updated clients.

- `400 – Failed To Update Client`

  The response contains details as to why the client couldn't be updated.

- `401 – Invalid Credentials`

  The user doesn't exist or isn't authorized to update a client.

- `500 – Internal Server Error`

  An unknown error has occurred.

GET

Retrieves details for all OAuth clients.

JSON Parameters

None

Return codes

- `200 – Success`

The body contains JSON data for all clients.

> **Note**
>
> The parameter `refreshRolling` isn't returned if the AS global setting is set for a client (the default).

- `400 – Failed To Retrieve Clients`

  The response contains details as to why clients couldn't be retrieved.

- `401 – Invalid Credentials`

  The user doesn't exist or isn't authorized.

- `500 – Internal Server Error`

  An unknown error has occurred.

Endpoint:`/pf-ws/rest/oauth/clients/<clientId>`

This resource accepts the GET and DELETE methods.

GET

Retrieves details for the specified client ID.

JSON Parameters

None.

Return codes

- `200 – Success`

  JSON client parameters are included.

> **Note**
>
> The parameter `refreshRolling` isn't returned if the AS global setting is set for a client, the default setting.

- `400 – Failed To Retrieve Client`

  The response contains details as to why client couldn't be retrieved.

- `401 – Invalid Credentials`

  The user doesn't exist or isn't authorized.

- `500 – Internal Server Error`

  An unknown error has occurred.

DELETE

Deletes records for the specified client ID.

JSON Parameters

None

Return codes

- `200 – Success`
- `400 – Failed To Delete Client`

  The response contains details as to why client couldn't be deleted.
- `401 – Invalid Credentials`

  The user doesn't exist or isn't authorized.
- `405 – Method Not Allowed`

  The client ID wasn't specified.
- `500 – Internal Server Error`

  An unknown error has occurred.

Logging

PingFederate records the actions performed through this endpoint in the `runtime-api.log` file. While the events themselves aren't configurable, you can adjust the `log4j2.xml` configuration settings to alter the format and desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method
- REST endpoint
- HTTP status code

Each of the above fields is separated by a vertical pipe (`\|`) for ease of parsing.

Related links

- PingFederate administrative API
- Grant types

- Asynchronous Front-Channel Logout
- Back-Channel Session Revocation
- Configuring authorization server settings
- Defining scopes
- Access token management
- OAuth client datastores
- Configuring OAuth clients

OAuth Access Grant Management Service

PingFederate includes a REST-based web service to manage OAuth persistent grants administratively.

The OAuth access grant management service enables retrieval and revocation of persistent grants and their associated extended attribute names and values, if any, on a per-client or per-resource owner basis.

> **Important**
>
> Client records must be stored in an external client data store.

> **Note**
>
> Applications must authenticate to this web service using HTTP Basic authentication and credentials validated through an instance of a Password Credential Validator (PCV). The PCV instance, in turn, must be selected in the OAuth authorization server (AS) configuration.

Endpoints

Manage by client

`/pf-ws/rest/oauth/clients/<clientId>/grants<grantId>]`

Manage by user (based on the USER_KEY value)

`/pf-ws/rest/oauth/users/<userKey>/grants<grantId>]`

Both REST resources accept the GET, for retrieval methods, and DELETE, for revocation methods. The results are formatted in JSON.

Parameters

Parameter	Description
`clientId`	The client ID for which to retrieve or revoke grants. Required to manage grants for a specific client.

Parameter	Description
`userKey`	The `USER_KEY` value for which to retrieve or revoke grants. Required to manage grants for a specific resource owner. **Tip** The `USER_KEY` value varies, depending on its fulfillment based on the mapping configuration defined in the **IdP Adapter Grant Mapping**, **Authentication Policy Contract Mapping**, or **Resource Owner Credentials Grant Mapping** tabs.
`grantId` (Optional)	The persistent grant identifier for which to retrieve or revoke a specific grant. The value corresponds to the value of the `id` field found in the JSON array of grants returned from a previous `GET /oauth/clients/<clientId>/grants` or `GET /oauth/users/<userKey>/grants` request. **Note** If this parameter is not specified, the request applies to all grants for the client or user.

Cross Site Request Forgery Protection

Both endpoints require the `X-XSRF-HEADER` HTTP Header with any value to prevent cross site request forgery.

Example 1

Sample request

A request to retrieve all grants for client with an ID value of `ac_client`. Note that this client is configured with Allowed Grant Types values of Authorization Code and Refresh Token.

```
GET /pf-ws/rest/oauth/clients/ac_client/grants HTTP/1.1
Host: localhost:9031
Authorization: Basic YWRtaW46MkZlZGVyYXRl
X-XSRF-HEADER: PingFederate
```

Sample response

```
{
    "items": [
        {
            "id": "ixqWL3k9ZnPxjTaphcFLrVqwdrtvc6tV",
            "userKey": "asmith",
            "grantType": "AUTHORIZATION_CODE",
            "scopes": [],
            "clientId": "ac_client",
            "issued": "2019-03-15T20:00:27.343Z",
            "updated": "2019-03-15T20:00:27.343Z",
            "grantAttributes": [
                {
                    "name": "pgeaAttrMulti",
                    "values": [
                        "CN=group1,OU=Resources,DC=example,DC=local",
                        "CN=group2,OU=Resources,DC=example,DC=local"
                    ]
                },
                {
                    "name": "pgeaAttrSingle",
                    "values": [
                        "asmith@example.local"
                    ]
                }
            ]
        }
    ]
}
```

Example 2

Sample request

A request to retrieve all grants for client with an ID value of `im_client`. This client is configured with an Allowed Grant Types value of Implicit, and PingFederate is configured to reuse existing persistent access grants for the implicit grant type on the System > OAuth Settings > [.wintitle] Authorization Server Settings** window.

```
GET /pf-ws/rest/oauth/clients/im_client/grants HTTP/1.1
Host: localhost:9031
Authorization: Basic YWRtaW46MkZlZGVyYXRl
X-XSRF-HEADER: PingFederate
```

Sample response

```
{
    "items": [
        {
            "id": "G7fV0HVALkplTl1Hdw109DYzlLNPXXIt",
            "userKey": "asmith",
            "grantType": "IMPLICIT",
            "scopes": [],
            "clientId": "im_client",
            "issued": "2019-03-15T18:20:39.652Z",
            "updated": "2019-03-15T18:20:39.652Z"
        }
    ]
}
```

For more information about persistent grants and their relationship with various grant types, see Transient grants and persistent grants.

Return codes

- 200 – Success
- 204 – Success with no content returned

  Returned when revoking a persistent grant.
- 401 – Invalid Credentials

  The user does not exist, the password is incorrect, or the user account is locked.
- 404 – Not Found

  Returned when the requested persistent grant or client is not found.
- 500 – Internal Server Error

  An unknown error has occurred.

Logging

PingFederate records the actions performed through this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you can adjust the `log4j2.xml` configuration settings to alter the format and desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method

- REST endpoint
- HTTP status code

Each of the above fields is separated by a vertical pipe (`\|`) for ease of parsing.

Related links

- Transient grants and persistent grants
- Password Credential Validators
- Configuring authorization server settings
- OAuth client datastores
- Configuring OAuth clients

OAuth Persistent Grant Management API

This REST-based runtime API facilitates the use case where clients can assume the responsibility of grant management if the users authorize the clients to do so.

In this scenario, a client prompts the user to approve a specific scope for managing persistent grants on the user's behalf. If the user approves, the client requests an access token with such scope from PingFederate. As long as the access token remains valid, the client can retrieve and revoke persistent grants and their associated extended attribute names and values, if any, for that user.

The activation of the Persistent Grant Management API requires two settings:

- The scope required to perform this task
- The Access Token Management instance under which the access tokens issued can be used by clients to manage persistent grants

You can find both settings from the System tab on the System > OAuth Settings > Authorization Server Settings window. Additionally, clients that are capable of managing grants on their users' behalf must be configured to use the selected Access Token Management instance in their client records.

> **Note**
>
> To use this runtime API, clients must authenticate by presenting a valid access token with the required scope. The token is presented as a bearer token through the HTTP Bearer authentication scheme.

> **Tip**
>
> Not all OAuth use cases involve persistent grants. For more information, see Transient grants and persistent grants.

Endpoint: /pf-ws/rest/oauth/grants<*grantId*>]

This REST resource accepts the GET for retrieval methods, and DELETE for revocation methods. The results are formatted in JSON.

Parameter

Parameter	Description
`grantId`	The persistent grant identifier to retrieve or revoke a specific grant. The value corresponds to the value of the `id` field found in the JSON array of grants returned by a previous GET request from this endpoint.

Cross Site Request Forgery Protection

This endpoint requires the `X-XSRF-HEADER` HTTP Header with any value to prevent cross-site request forgery.

Sample request

A request to retrieve all grants.

```
GET /pf-ws/rest/oauth/grants/ HTTP/1.1
Host: localhost:9031
Authorization: Bearer eyJhbG...IKqMfg
X-XSRF-HEADER: PingFederate
```

In this example, the client prompted the resource owner (Joe) to authorize it to manage persistent grants on his behalf. Joe agreed and approved the requested `admin` scope. The client then obtained an access token with the scope from PingFederate. The issued access token is a self-contained JSON web token (JWT).

This sample request illustrates a GET request from the client. The client wants to retrieve a list of grants associated with Joe by presenting the access token to the Persistent Grant Management API for authentication. The access token is truncated for readability.

+

Note

To use a demonstrating proof-of-possession (DPoP)-bound access token, use the DPoP authentication scheme in the Authorization header and include a DPoP Proof JWT in the DPoP header. For more information, see the OAuth 2.0 Demonstrating Proof-of-Possession at the Application Layer specification and the description of the PingFederate DPoP settings in Configuring authorization server settings.

Sample response

```
{
    "items": [
        {
            "id": "5a4nszZOppgo9RfRtrVXNY0Eq5ka1YZ6",
            "userKey": "joe",
            "grantType": "RESOURCE_OWNER_CREDENTIALS",
            "scopes": [
                "phone"
            ],
            "clientId": "ro_client",
            "issued": "2018-12-15T00:54:30.190Z",
            "updated": "2018-12-15T00:54:30.190Z"
        },
        {
            "id": "PTfURLoaXC97JXU6uilAORSkFQoMOLyV",
            "userKey": "joe",
            "grantType": "AUTHORIZATION_CODE",
            "scopes": [
                "openid",
                "profile",
                "admin"
            ],
            "clientId": "ac_oic_client",
            "issued": "2018-12-15T18:29:26.018Z",
            "updated": "2018-12-15T18:29:26.018Z"
        },
        {
            "id": "k1oFbxujlGHbfEBfVqDj0aIDllFBzghX",
            "userKey": "joe",
            "grantType": "AUTHORIZATION_CODE",
            "scopes": [
                "openid",
                "email",
                "phone"
            ],
            "clientId": "ac_oic_client",
            "issued": "2018-12-15T19:03:29.439Z",
            "updated": "2018-12-15T19:03:29.439Z"
        },
        {
            "id": "EhzZYggNuFT9EWpw9p21SYCddwAPFAKR",
            "userKey": "joe",
            "grantType": "IMPLICIT",
            "scopes": [
                "openid",
                "profile"
            ],
            "clientId": "im_oic_client",
            "issued": "2018-12-15T19:04:05.044Z",
            "updated": "2018-12-15T19:04:05.044Z"
        }
    ]
}
```

In this example, PingFederate returned four persistent grants associated with the resource owner from three clients.

Return codes

- `200` – Success
- `204` – Success with no content returned

  Returned when revoking a persistent grant.
- `401` – Invalid Credentials

  The access token is invalid (including the lack of the required scope) or missing.
- `404` – Not Found

  Returned when the requested persistent grant is not found or a grant ID is missing.
- `500` – Internal Server Error

An unknown error has occurred.

Logging

PingFederate records the actions performed through this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you can adjust the `log4j2.xml` configuration settings to alter the format and desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method
- REST endpoint
- HTTP status code

Each of the above fields is separated by a vertical pipe (`\|`) for ease of parsing.

Related links

- Configuring authorization server settings
- Configuring OAuth clients

Session Management API by session identifiers

The runtime session management application programming interface (API) lets OAuth clients get information about user sessions, extend sessions, revoke sessions, and delete sessions. Knowing the remaining lifetime of a valid session could, for example, let an application prompt the user to extend the session.

An OAuth client can take the pi.sri session identifier from its access token and send it to the session management API in an HTTP GET request. An OAuth client can also send the pi.sri to the session management API in an HTTP POST request to extend or revoke the sessions. The API returns information in JavaScript Object Notation (JSON) format about each session associated with the pi.sri, including:

- Last activity time
- Idle and maximum timeout time
- Authentication source that created the session
- Context data
  - IP address
  - User agent

> **Note**
>
> The API response body includes only sessions that were configured with the authentication sessions capability described in Configuring authentication sessions.

The session management API works with sessions stored in persistent storage and across clustered nodes. For this API, the runtime APIs audit log only records session revoke events.

> **Important**
>
> OAuth clients must authenticate to the API using their configured client authentication method.

To configure PingFederate so that an OAuth client can use the session management API:

- Allow the client to access the session management API, as described in Configuring OAuth clients.
- Configure the client's access token manager instance to include a pi.sri in access tokens, as described in Managing session validation settings.

Session management API by session identifiers endpoints

The session management API by session identifiers has three endpoints, which all require the `sri` parameter.

The OpenID Provider configuration endpoint `/.well-known/openid-configuration` provides configuration information for OAuth clients to access the session management API endpoints. For more information, see OpenID Provider configuration endpoint.

The session management API endpoints return several response codes, including:

- `200 OK: The request was successfully processed.`
- `400 Bad request: The format of the SRI is invalid.`

Endpoint `/pf-ws/rest/sessionMgmt/sessions/{sri}`

Use HTTP GET requests to get information about all sessions associated with the pi.sri specified by the `sri` parameter.

Here is a sample GET request from a client to the `/pf-ws/rest/sessionMgmt/sessions/{sri}` endpoint:

```
GET /pf-ws/rest/sessionMgmt/sessions/qzTEiEroxdzAufjYKQawm72lcBE..4RbA HTTP/1.1
Host: www.example.com
X-XSRF-Header: PingFederate
Authorization: Basic YWNfY2xpZW50OmdPWDh0NEQ...h3ZjI=
Cookie: PF=K60mOoBlTvWcD4frFzcKF5
```

After receiving a successful request, the endpoint returns a response like one of the following samples, depending on whether the status is `HAS_VALID_SESSIONS`, `NO_VALID_SESSIONS`, or `SESSION_REVOKED`:

```
{
    "sri": "qzTEiEroxdzAufjYKQawm72lcBE..4RbA",
    "status": "HAS_VALID_SESSIONS",
    "lastActivityTime": "2020-06-10T17:25:00.461Z",
    "authnSessions": [ // This section can include multiple sessions
        {
            "authnSource": {
                "sourceType": "IDP_CONN",
                "id": "L07d8fse7dslShd6d_20HA8jP6",
                "entityId": "Amazon_Africa_A" // Only for IDP_CONN sourceType sessions
            },
            "id": "ba5a3d97afee5ef9450b710ff932680e3579dc7f",
            "creationTime": "2020-06-10T17:25:00.454Z",
            "idleTimeout": "2020-06-10T18:25:00.461Z",
            "maxTimeout": "2020-06-11T01:25:00.461Z"
        },
        {
            "authnSource": {
                "sourceType": "ADAPTER",
                "id": "HtmlFormAdapter",
                "adapterType": "HTML Form IdP Adapter" // Only for ADAPTER sourceType sessions
            },
            "id": "7cbef5022be8d841f14a95ace8987cbb34c77a21",
            "creationTime": "2020-06-10T17:25:00.454Z",
            "idleTimeout": "2020-06-10T18:25:00.461Z",
            "maxTimeout": "2020-06-11T01:25:00.461Z"
        }
    ],
    "contextData": {
        "ipAddress": "127.0.0.1",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
94.0.4606.81 Safari/537.36"
    }
}
```

```
{
    "sri": "qzTEiEroxdzAufjYKQawm72lcBE..4RbA",
    "status": "NO_VALID_SESSIONS",
}
```

```
{
    "sri": "qzTEiEroxdzAufjYKQawm72lcBE..4RbA",
    "status": "SESSION_REVOKED",
}
```

Endpoint `/pf-ws/rest/sessionMgmt/sessions/{sri}/extend`

Use HTTP POST requests to extend the `idleTimeout` value of all sessions associated with the pi.sri specified by the `sri` parameter.

Here is a sample POST request from a client to the `/pf-ws/rest/sessionMgmt/sessions/{sri}/extend` endpoint:

```
POST /pf-ws/rest/sessionMgmt/sessions/qzTEiEroxdzAufjYKQawm72lcBE..4RbA/extend HTTP/1.1
Host: www.example.com
X-XSRF-Header: PingFederate
Authorization: Basic YWNfY2xpZW50OmdPWDh0NEQ...h3ZjI=
Cookie: PF=K60mOoBlTvWcD4frFzcKF5
```

After receiving a successful request, the endpoint returns a response like the following sample:

```
{
    "sri": "Y9tTHRVD7s55Vn2hdrrVxWgRD44..aHbY.tEXnpsf6V1YGT3OWMNWxZjp2m",
    "status": "HAS_VALID_SESSIONS",
    "lastActivityTime": "2021-10-14T18:29:00.195Z",
    "authnSessions": [
        {
            "authnSource": {
                "sourceType": "ADAPTER",
                "id": "CIAMHtml",
                "adapterType": "HTML Form IdP Adapter"
            },
            "id": "8e95ab8600d71a6091af61d54b75ddefb9270c2c",
            "creationTime": "2021-10-14T18:28:47.205Z",
            "idleTimeout": "2021-10-14T19:29:00.195Z",
            "maxTimeout": "2021-10-15T02:28:47.205Z"
        }
    ]
    "contextData": {
        "ipAddress": "127.0.0.1",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
94.0.4606.81 Safari/537.36"
    }
}
```

Endpoint `/pf-ws/rest/sessionMgmt/sessions/{sri}/revoke`

Use HTTP POST requests to revoke all sessions associated with the pi.sri specified by the `sri` parameter.

Here is a sample POST request from a client to the `/pf-ws/rest/sessionMgmt/sessions/{sri}/revoke` endpoint:

```
POST /pf-ws/rest/sessionMgmt/sessions/qzTEiEroxdzAufjYKQawm72lcBE..4RbA/revoke HTTP/1.1
Host: www.example.com
X-XSRF-Header: PingFederate
Authorization: Basic YWNfY2xpZW50OmdPWDh0NEQ...h3ZjI=
Cookie: PF=K60mOoBlTvWcD4frFzcKF5
```

After receiving a successful request, the endpoint returns a response like the following sample:

```
{
    "sri": "qzTEiEroxdzAufjYKQawm72lcBE..4RbA",
    "status": "SESSION_REVOKED",
}
```

Endpoint `/pf-ws/rest/sessionMgmt/sessions/{sri}/authnSessions/{id}`

Use HTTP DELETE requests to remove an authentication session with the specified `id` parameter from the sessions associated with the pi.sri specified by the `sri` parameter.

Here is a sample DELETE request from a client to the `/pf-ws/rest/sessionMgmt/sessions/{sri}/authnSessions/{id}` endpoint:

```
DELETE /pf-ws/rest/sessionMgmt/sessions/qzTEiEroxdzAufjYKQawm72lcBE..4RbA/authnSessions/
ba5a3d97afee5ef9450b710ff932680e3579dc7f HTTP/1.1
Host: www.example.com
X-XSRF-Header: PingFederate
Authorization: Basic YWNfY2xpZW50OmdPWDh0NEQ...h3ZjI=
Cookie: PF=K60mOoBlTvWcD4frFzcKF5
```

After receiving a successful request, the endpoint returns an HTTP status of `204 No Content`.

Session Management API by user identifiers

OAuth applications can send a user identifier to the Session Management API endpoint to query or revoke authentication sessions belonging to the end-user.

> **Note**
>
> - **Allow Access to Session Management API** must be enabled on an OAuth client to allow the client to access this session management API by user identifier.
> - Adapters must be configured with a Unique User Key Attribute, as described in Setting pseudonym and masking options.
> - The `user_key` specified in the session management API request must match the value of the Unique User Key Attribute determined at runtime.
> - This API is different than the other session management APIs in that it only returns currently valid sessions (status = HAS_VALID_SESSIONS). Revoked sessions (SESSION_REVOKED) or users without any sessions (NO_VALID_SESSIONS) will not be seen from this API.
> - The `user_key` parameter needs to be URL encoded (@ is encoded as %40) since the value appears as a query parameter in the URL.

Applications can query or revoke user sessions based on user identifiers. The bulk revocation capability provides a convenient way to expire server-side authentication sessions on a per-user basis if needed. When revoked and without valid credentials, such end-users will not be able to fulfill authentication requirements and access protected resources.

An OAuth client needs to obtain a user identifier and send it to the session management API in an HTTP GET or POST request. If the client is configured to obtain an access token, an ID token, or a User Info response, it could potentially get the user identifier that way. However, if the client is not configured to obtain one of these items, or if none of them delivers the applicable user identifier, the client can still use this endpoint by sending a user identifier. For example, an application for the help desk to query or revoke all authentication sessions for a given user falls into this category. In this example, the developer will build an application to prompt the help desk staff to provide a user identifier, which becomes the `user_key` parameter value, send a GET or POST request to the session management API endpoint, and display the result of the request.

The API returns information in JSON format about each session associated with the user, such as:

- SRI
- Status
- Last activity time
- Authentication sessions
    - Authentication Source
    - ID of the per-authentication source session information
    - Creation time
    - Idle and maximum timeout
- Context data
    - IP address
    - User agent

> **Note**
>
> The API response body includes only sessions that were configured with the authentication sessions capability described in Configuring authentication sessions.

An OAuth client can also send the user identifier to the session management API in an HTTP POST request to revoke the sessions.

The session management API works with sessions stored in persistent storage and across clustered nodes. For this API, the runtime API's audit log only records session revoke events.

> **Important**
>
> OAuth clients must authenticate to the API using their configured client authentication method.

To configure PingFederate so that an OAuth client can use the session management API, allow the client to access the session management API, as described in Configuring OAuth clients

Session management API by user identifier endpoints

The session management API by user identifiers has one endpoint, which requires the `user_key` parameter.

The OpenID Provider configuration endpoint `/.well-known/openid-configuration` provides configuration information for OAuth clients to access the session management API endpoints. For more information, see OpenID Provider configuration endpoint.

Endpoint `/pf-ws/rest/sessionMgmt/users/{user_key}`

Use HTTP GET requests to get information about all sessions associated with the user specified by the `user_key` parameter.

Sample request

```
GET /pf-ws/rest/sessionMgmt/users/john%40test.com-east HTTP/1.1
```

Sample response

```
{
    "sri": "qzTEiEroxdzAufjYKQawm72lcBE..4RbA",
    "status": "HAS_VALID_SESSIONS",
    "lastActivityTime": "2020-06-10T17:25:00.461Z",
    "authnSessions": [ // This section can include multiple sessions
        {
            "authnSource": {
                "sourceType": "IDP_CONN",
                "id": "L07d8fse7dslShd6d_20HA8jP6",
                "entityId": "Amazon_Africa_A" // Only for IDP_CONN sourceType sessions
            },
            "id": "ba5a3d97afee5ef9450b710ff932680e3579dc7f",
            "creationTime": "2020-06-10T17:25:00.454Z",
            "idleTimeout": "2020-06-10T18:25:00.461Z",
            "maxTimeout": "2020-06-11T01:25:00.461Z"
        },
        {
            "authnSource": {
                "sourceType": "ADAPTER",
                "id": "HtmlFormAdapter",
                "adapterType": "HTML Form IdP Adapter" // Only for ADAPTER sourceType sessions
            },
            "id": "7cbef5022be8d841f14a95ace8987cbb34c77a21",
            "creationTime": "2020-06-10T17:25:00.454Z",
            "idleTimeout": "2020-06-10T18:25:00.461Z",
            "maxTimeout": "2020-06-11T01:25:00.461Z"
        }
    ],
    "contextData": {
        "ipAddress": "127.0.0.1",
        "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/94.0.4606.81 Safari/537.36"
    }
}
```

Endpoint /pf-ws/rest/sessionMgmt/users/{user_key}/revoke

Use HTTP POST requests to revoke all PingFederate runtime authentication sessions that are associated with the given user identifier `user_key`. After running this request, any revoked sessions will be logged with an event as "SRI_REVOKED" in the audit log.

Sample request

```
POST /pf-ws/rest/sessionMgmt/users/john%40test.com-east/revoke HTTP/1.1
Host: www.example.com
X-XSRF-Header: PingFederate
Authorization: Basic YWNfY2xpZW50OmdPWDh0NEQ...h3ZjI=
Cookie: PF=K60mOoBlTvWcD4frFzcKF5
```

Return codes

- 200 OK: The request was successfully processed
- 503 Service Unavailable: An error occurred while deleting stored sessions

> **Note**
>
> You can optionally use the endpoints described in Session Management API by session identifiers if needed.

Session Revocation API endpoint

PingFederate includes a REST-based web service for back-channel session revocation.

This service enables OAuth clients to add sessions to the revocation list or to query their revocation status. This endpoint is not part of the OAuth specification. You must select the Allow Access to Session Revocation API check box in the configuration for the applicable clients. This endpoint is a URL path extension of the PingFederate runtime endpoint. For example, https://www.example.com:9031/pf-ws/rest/sessionMgmt/revokedSris.

> **Important**
>
> OAuth clients must authenticate to the web service using their configured client authentication method. For more information about OAuth client authentication methods, see Configuring OAuth clients.

> **Note**
>
> If clients are allowed to add sessions to the revocation list, you can enable the **Check session revocation status** option in the applicable Access Token Management instances for the token validation process to consider whether a session has been added to the revocation list

Endpoint: `/pf-ws/rest/sessionMgmt/revokedSris`

This resource accepts the POST and GET methods. It also requires the `X-XSRF-HEADER` HTTP header with any value to prevent cross site request forgery.

> **Note**
>
> The POST method described in this section requires the element name/value pair formatted in JSON.

> **Tip**
>
> You can find information about the Session Revocation API endpoint in the OpenID Provider Configuration endpoint metadata: `ping_revoked_sris_endpoint` .

POST

A POST request adds a session to the revocation list based on its session identifier, `id`, in the POST data. The ID value corresponds to that of the `pi.sri` element in the ID token. The required `Content-Type` is `application/json`.

Sample request

A POST request to add a session with a session identifier of abc123 to the revocation list.

```
POST /pf-ws/rest/sessionMgmt/revokedSris
Host: localhost:9031
Authorization: Basic
YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT25tbGlqaG9hdXRocGxheWdy
X-XSRF-HEADER: PingFederate
Content-Type: application/json

{"id":"abc123"}
```

Return codes

- 201 – Created

  The session is added to the revocation list.
- 400 – Bad Request

  The `X-XSRF-HEADER` HTTP header is not found in the HTTP POST request.
- 401 – Unauthorized

  The response contains details as to why the attempt failed.
- 415 – Unsupported Media Type

  The `Content-Type: application/json` HTTP header is not found in the HTTP POST request.
- 500 – Internal Server Error

  An unknown error has occurred.

GET

A GET request sends a query for the revocation status for a session with its session identifier, `id`, appended to the endpoint. The ID value corresponds to that of `pi.sri` in the ID token.

Sample request

A GET request to query the revocation status for a session with a session identifier of abc123.

```
GET /pf-ws/rest/sessionMgmt/revokedSris/abc123
Host: localhost:9031
Authorization: Basic
YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT25tbGlqaG9hdXRocGxheWdy
X-XSRF-HEADER: PingFederate
```

If PingFederate authentication sessions are enabled, querying a valid session also extends the session lifetime by the time value specified in the global Idle Timeout field or the idle timeout override for the authentication source associated with the session. The latter takes precedence. For externally stored authentication sessions, this operation is optimized to only send updates to the external storage when the remaining idle timeout window is less than 75%.

Alternatively, include the optional `updateActivityTime` query parameter and set the value to `false` in the request to query the status of a session without extending its lifetime.

Example

```
GET /pf-ws/rest/sessionMgmt/revokedSris/abc123?updateActivityTime=false
Host: localhost:9031
Authorization: Basic
YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT25tbGlqaG9hdXRocGxheWdyb3VuZGFwcGxpY
X-XSRF-HEADER: PingFederate
```

Return codes

- 200 – OK

  ```
  [.codeph]``\{"id":"abc123"}`` is found in the revocation list.
  ```

- 400 – Bad Request

  The `X-XSRF-HEADER` HTTP header is not found in the HTTP POST request.

- 401 – Unauthorized

  The response contains details as to why the attempt failed.

- 404 – Not Found

  ```
  [.codeph]``\{"resultId":"session_mgmt_sri_not_revoked","message":"The SRI has
                               not been revoked."}`` if the session is not found in the revocation
  list. If {pingfed} is configured to manage authentication sessions and the request does not come
  with the [.codeph]``updateActivityTime=false`` query parameter, the session is extended as well.
  ```

- 500 – Internal Server Error

  An unknown error has occurred.

Logging

PingFederate records the actions performed through this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you can adjust the `log4j2.xml` configuration settings to alter the format and desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP

- HTTP method
- REST endpoint
- HTTP status code

Each of the above fields is separated by a vertical pipe (`\|`) for ease of parsing.

Related links

- Back-Channel Session Revocation
- Sessions

PingFederate administrative API

PingFederate includes a REST-based application programming interface (API) for administrative functions. The administrative API provides a programmatic way to make configuration changes to PingFederate as an alternative to using the administrative console.

The configuration changes that you can make through the administrative API include, but are not limited to:

- Adapters and connections
- Authentication policy contracts
- Cluster management
- Data stores and password credential validators
- Keys and certificates
- License management
- Local administrative account management
- OAuth settings
- Server settings

For a complete list, see Accessing the API interactive documentation.

Initial setup using the administrative API

After installing PingFederate you can make four unauthenticated administrative API requests to perform the following tasks:

1. A GET request to `/license/agreement` to retrieve a URL to the license agreement.
2. A PUT request to `/license/agreement` to accept the license agreement.
3. A PUT request to `/license` to import a license file.
4. A POST request to `/administrativeAccounts` to create the first local administrative account, for native authentication.

You must assign the User Admin administrative role, `USER_ADMINISTRATOR`, to the first local administrative account. Other administrative roles are optional at this point. For more information, see the interactive documentation for the administrative API Accessing the API interactive documentation.

After the first local administrative account is created, you can make other authenticated administrative API requests to configure various components in PingFederate.

Authentication and authorization

Similar to the administrative console, access to the administrative API is protected after initial setup. The administrative API supports various authentication and authorization options. For more information, see Configure access to the administrative API.

Concurrent access

The administrative API supports concurrent access. When concurrent API calls are made to modify the same API resource, such as the identity provider (IdP) adapter instance or the service provider (SP) connection, PingFederate processes the last request made.

Logging

PingFederate records actions performed through the administrative API in the `admin-api.log` file. Information includes the time of the event, the action performed, the authentication method, and other fields. For more information, see Administrative API audit log.

Configure access to the administrative API

Similar to the administrative console, access to the administrative API after initial setup might be protected by several authentication and authorization schemes.

Access to the administrative API after initial setup is protected by one of the following authentication and authorization schemes:

- Native authentication, against local administrative accounts
- LDAP authentication
- RADIUS authentication
- Mutual TLS client certificate-based authentication
- OAuth 2.0 authorization

For new installations, native authentication is the default.

For upgrades, if the authentication or authorization method of the administrative API was not previously set, such as when upgrading from PingFederate 7.3 or an earlier version, the Upgrade Utility sets the value to that of the administrative console. Otherwise, it preserves the previously set value, such as when upgrading from PingFederate 8.0 to a future release.

The authentication or authorization method for the administrative API can change at a later time to any of the choices, regardless of which authentication or authorization method is chosen for the administrative console.

Besides authentication and authorization, PingFederate also provides role-based access control, as shown in the following table. The roles assigned to the accounts affect the results of the API calls.

PingFederate User Access Control

Account type	Administrative role	Access privileges
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Admin	Admin	Configure partner connections and most system settings, except the management of local accounts and the handling of local keys and certificates.
Admin	Expression Admin	Map user attributes by using the expression language, Object-Graph Navigation Language (OGNL). **Important** Only Administrative users who have both the Admin role and the Expression Admin role: • Can be granted the User Admin role. This restriction prevents non-Expression Admin users from granting themselves the Expression Admin Role. • Can be granted write access to the file system or directory where PingFederate is installed. This restriction prevents a non-Expression Admin user from placing a `data.zip` file containing expressions into the `<pf_install>/pingfederate/server/default/deploy` directory, which would introduce expressions into PingFederate.
Admin	Crypto Admin	Manage local keys and certificates.
Auditor	Not applicable	View-only permissions for all administrative functions. When the Auditor role is assigned, no other administrative roles can be set.

> **Note**
>
> All four administrative roles are required to access and make changes through the following services:
>
> - The `/bulk`, `/configArchive`, and `/configStore` administrative API endpoints
> - The **Configuration Archive** window, accessed from **System > Server**, in the administrative console
> - The **Connection Management** configuration item on the **Service Authentication** window, accessed from **Security > System Integration**

Enabling native authentication for the administrative API

When the administrative API is protected by native authentication, access to the administrative API is restricted to the users defined in the Account Management window.

About this task

The API calls must be authenticated by valid credentials over HTTP Basic authentication; otherwise, the administrative API returns an error message. The roles assigned to the users affect the results of the API calls.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, set the value of the `pf.admin.api.authentication` property to `native`. Then restart PingFederate.

   > **Note**
   >
   > You can configure PingFederate to support both `native` authentication and OAuth 2.0 authorization by specifying two values separated with a comma. For example, specify `pf.admin.api.authentication=OAuth2,native`. Supporting two authentication methods is helpful when you want to change applications from one method to another. For more information about supporting two authentication methods, see the description of `pf.admin.api.authentication` in Configuring PingFederate properties.

   > **Note**
   >
   > In a clustered PingFederate environment, you only need to modify `run.properties` on the console node.

2. Sign on to the administrative console with an account that has the User Admin role.

   > **Important**
   >
   > When the administrative console is protected by an alternative console authentication, such as certificate-based, LDAP, or RADIUS authentication, most user-management functions are handled outside the scope of the PingFederate administrative console. Therefore, the administrative console disables the functionality of the **System** > **Server** > **Administrative Accounts** window unless the logged-on administrator has been granted User Admin permissions.
   > To create or manage users in this scenario, add at least one external account to the role setting `userAdmin` in the configuration file for the respective authentication method. When the administrator logs on to the administrative console, the **Administrative Accounts** window becomes available to create or manage users for the purposes of accessing the administrative API.
   > For more information about the alternative console authentication and the respective configuration, see Alternative console authentication.

3. On the Administrative Accounts window, create or manage users as needed, and assign various PingFederate administrative roles as indicated by the PingFederate User Access Control table. For more information, see Configure access to the administrative API.

   > **Note**
   >
   > When assigning roles, remember that all users defined in the **Administrative Accounts** window can access the administrative API and the administrative console.

Enabling LDAP authentication

When the administrative application programming interface (API) is protected by Lightweight Directory Access Protocol (LDAP) authentication, the API calls must be authenticated by valid LDAP credentials over HTTP Basic authentication; otherwise, the administrative API returns an error message.

About this task

The LDAP authentication setup, including role assignment, is available through `<pf_install>/pingfederate/bin/ldap.properties`. The roles assigned to the LDAP accounts affect the results of the API calls.

> **Note**
>
> When you configure LDAP authentication, PingFederate does not lock out accounts based upon the number of failed sign-on attempts. The LDAP server is responsible for preventing access and is enforced according to its password lockout settings.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, set the value of the `pf.admin.api.authentication` property to `LDAP`.

   > **Note**
   >
   > You can configure PingFederate to support both `LDAP` authentication and OAuth 2.0 authorization by specifying two values separated with a comma. For example, specify `pf.admin.api.authentication=OAuth2,LDAP`. Supporting two authentication methods is helpful when you want to change applications from one method to another. For more information about supporting two authentication methods, see the description of `pf.admin.api.authentication` in Configuring PingFederate properties.

2. In the `<pf_install>/pingfederate/bin/ldap.properties` file, change property values as needed for your network configuration. For instructions and additional information, see the comments in the file.

   > **Important**
   >
   > Remember to assign LDAP users or designated LDAP groups, or both, to at least one of the PingFederate administrative roles, as indicated in the properties file. For information about permissions attached to the PingFederate roles, see the PingFederate User Access Control table in Configure access to the administrative API.

   > **Note**
   >
   > When you assign roles, remember that all LDAP accounts specified in `ldap.properties` can access the administrative API and the administrative console.

   > **Tip**
   >
   > You can also use this configuration file in conjunction with RADIUS authentication to determine permissions dynamically with an LDAP connection.

3. Restart PingFederate.

> **Note**
>
> In a clustered PingFederate environment, you only need to modify `run.properties` and `ldap.properties` on the console node.

Enabling RADIUS authentication

The RADIUS protocol provides a common approach for implementing strong authentication in a client-server configuration.

About this task

The RADIUS authentication setup is available through configuration files in the `<pf_install>/pingfederate/bin` directory. The administrative API supports the protocol scenario for one-step authentication, for example, appending a one time password (OTP) after the password.

When RADIUS authentication is protecting the administrative API, the API calls must be authenticated by valid credentials over HTTP Basic authentication. Otherwise, the administrative API returns an error message. The roles assigned to the accounts affect the results of the API calls.

> **Note**
>
> When you configure RADIUS authentication, PingFederate does not lock out accounts based upon the number of failed logon attempts. Instead, responsibility for preventing access is delegated to the RADIUS server and enforced according to its password lockout settings.

> **Note**
>
> The NAS-IP-Address attribute is added to all Access-Request packets sent to the RADIUS server. The value is copied from the `pf.engine.bind.address` property in `run.properties`. Only IPv4 addresses are supported.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, set the value of the `pf.admin.api.authentication` property to `RADIUS`.

   > **Note**
   >
   > You can configure PingFederate to support both `RADIUS` authentication and OAuth 2.0 authorization by specifying two values separated with a comma. For example, specify `pf.admin.api.authentication=OAuth2,RADIUS`. Supporting two authentication methods is helpful when you want to change applications from one method to another. For more information about supporting two authentication methods, see the description of `pf.admin.api.authentication` in Configuring PingFederate properties.

2. In the `<pf_install>/pingfederate/bin/radius.properties` file, change property values as needed for your network configuration. For instructions and additional information, see the comments in the file.

> **Important**
>
> Assign RADIUS users or designated RADIUS groups, or both, to at least one of the PingFederate administrative roles as indicated in the properties file. Alternatively, you can set the `use.ldap.roles` property to `true` and use the LDAP properties file, which is also in the `bin` directory, to map LDAP group-based permissions to PingFederate roles. For more information about permissions attached to the PingFederate roles, see the PingFederate User Access Control table in Configure access to the administrative API.

> **Note**
>
> When assigning roles, remember that all accounts specified in `radius.properties` can access the administrative API and the administrative console.

3. Restart PingFederate.

> **Note**
>
> In a clustered PingFederate environment, you only need to modify `run.properties` and `radius.properties` on the console node.

Enabling certificate-based authentication

When client-certificate authentication is enabled, the API calls must be authenticated by X.509 client certificates; otherwise, the administrative API returns an error message.

About this task

In addition to X.509 client certificate authentication, the corresponding root certificate authority (CA) certificates must either be contained in the Java runtime or be imported into the PingFederate's Trusted CA store. For more information, see Manage trusted certificate authorities.

The rest of the certificate-based authentication setup, including specifying the Issuer DN of the root CA certificates and the applicable roles of the client certificates, is available through `<pf_install>/pingfederate/bin/cert_auth.properties`. The roles assigned to the certificates affect the results of the API calls.

Steps

1. Sign on to the administrative console with an account that has the role Crypto Admin.
2. Ensure the client-certificate's root CA and any intermediate CA certificates are contained in the trusted store, either for the Java runtime, or PingFederate, or both.

> **Note**
>
> To import a certificate, click **Trusted CAs** in the Certificate Management section under Server Configuration.

> **Tip**
>
> Click the Serial number and copy the Issuer distinguished name (DN) to use in a couple steps later.

3. Verify the `pf.admin.api.authentication` value in `<pf_install>/pingfederate/bin/run.properties` is set to `cert`. Update as needed.

4. In the `<pf_install>/pingfederate/bin/cert_auth.properties` file, enter the Issuer DN for the client certificate as a value for the property: `rootca.issuer.<x>`, where *<x>* is a sequential number starting at 1. For more information, see the properties file.

   > **Important**
   >
   > The configuration values are case-sensitive.

   If you copied the Issuer DN a couple steps earlier, paste this value.

5. Repeat the previous step for any additional CAs as needed.

6. Enter the certificate's Subject DN for the applicable PingFederate permission roles, as described in the properties file. For information about permissions attached to the PingFederate roles, see the PingFederate User Access Control table in Configure access to the administrative API.

   > **Important**
   >
   > The configuration values are case-sensitive.

   > **Note**
   >
   > When assigning roles, keep in mind that all client certificates specified in `cert_auth.properties` can be used to access the administrative API and the administrative console.

7. Repeat the previous step for all client certificates as needed.

8. Restart PingFederate.

   > **Note**
   >
   > In a clustered PingFederate environment, you only need to modify `run.properties` and `cert_auth.properties` on the console node.

Enabling OAuth 2.0 authorization

PingFederate clients can gain access to the administrative API endpoint by providing an OAuth 2.0 access token. The `<pf_install>/pingfederate/bin/oauth2.properties` file contains settings that allow you to configure information required to interact with the authorization server as a client.

Steps

1. In the `<pf_install>/pingfederate/bin/run.properties` file, set the value of the `pf.admin.api.authentication` property to `OAuth2`.

> **Note**
>
> You can also configure PingFederate to support both `OAuth2` authorization and a basic authentication method by specifying two values separated with a comma. For example, specify `pf.admin.api.authentication=OAuth2,LDAP`. The basic authentication methods are `native`, `LDAP`, and `RADIUS`. Supporting two authentication methods is helpful when you want to change applications from one method to another. For more information about supporting two authentication methods, see the description of `pf.admin.api.authentication` in Configuring PingFederate properties.

2. In the `<pf_install>/pingfederate/bin/oauth2.properties` file, change property values as needed. For instructions and additional information, see the comments in the file.

> **Important**
>
> Remember to assign at least one of the PingFederate administrative roles, as indicated in the properties file. For information about permissions attached to the PingFederate roles, see the PingFederate User Access Control table in Configure access to the administrative API.

3. Restart PingFederate.

> **Note**
>
> In a clustered PingFederate environment, you only need to modify `run.properties` and `oauth2.properties` on the console node.

Accessing the API interactive documentation

PingFederate ships with interactive documentation for both developers and non-developers to explore the API endpoints, view documentation for the API, and experiment with API calls.

About this task

In general, you can make API calls from an interactive user interface, custom applications, or from command line tools such as cURL. The endpoint is only available at the administrative port, as defined by the `pf.admin.https.port` property in `<pf_install>/pingfederate/bin/run.properties`.

> **Important**
>
> For enhanced API security, you must include `X-XSRF-Header: PingFederate` in all requests and use the `application/json` content type for PUT and POST requests.

To access the administrative API documentation, follow these steps:

Steps

1. Start PingFederate.

2. Start a web browser.

3. Browse to the following URL: https://<*pf_host*>:9999/pf-admin-api/api-docs/

> **Note**
>
> *<pf_host>* is the network address of your PingFederate server. It can be an IP address, a host name, or a fully qualified domain name. It must be reachable from your computer.
> `9999` is the default value of the `pf.admin.https.port` property in the `run.properties` file.

> **Tip**
>
> The administrative API is also documented in the OpenAPI Specification, previously known as the Swagger Specification. Click on the `/pf-admin-api/v1/swagger.json` URL on the Administrative API Documentation page to access the contents.

4. To test an administrative API, follow these steps:

   1. Select a section of the administrative API you would like to explore; for example, /dataStores.
   2. Expand the method you want to use; for example, GET /dataStores.
   3. Enter required parameters, if any. For more information, see Operation Models underneath the selected API endpoint.
   4. Click Try it out.

      > **Note**
      >
      > You might be prompted to sign on using administrative credentials over HTTP Basic authentication. The role assigned to the respective administrative accounts affects the access to the requested API.

      Result:

      If the request completes successfully, the administrative API returns the Request URL, the Response Body, the Response Code, and the Response Headers.

Application endpoints

Application endpoints provide a means, through standard HTTP, by which external applications can communicate with the PingFederate server.

The single sign-on (SSO) and single log-out (SLO) endpoints for an identity provider (IdP) and a service provider (SP) include optional parameters which you can use to specify error pages that users see in the event of an SSO or SLO failure. By default, PingFederate provides templates for these and other errors or conditions. For more information, see Customizable user-facing pages.

SP endpoints also include those available for system for cross-domain identity management (SCIM) inbound provisioning. For more information, see Provisioning for SPs.

For either SP or IdP servers, PingFederate provides a maintenance endpoint for administrators to verify that the server is running. Endpoints applicable to both server roles include those needed for adapter-to-adapter mapping and retrieval of WS-Trust metadata. For more information, see Adapter-to-adapter mappings and WSC and WSP support.

PingFederate provides a favorite icon for all application endpoints. For more information, see Customizing the favicon for application and protocol endpoints.

IdP endpoints

The following sections describe PingFederate identity provider (IdP) endpoints, including the case-sensitive query parameters that each accepts or requires. These endpoints accept either the HTTP GET or POST methods.

Begin each URL with the fully-qualified server name and port number of your PingFederate IdP server, for example, `https://www.example.com:9031/idp/startSSO.ping`.

> **Important**
>
> When using the parameters `TargetResource` or `TARGET` with their own query parameters included, the parameter value must be URL-encoded. Any other parameters that contain restricted characters, such as many SAML URNs, also must be URL-encoded. For information about URL encoding, see third party resources such as HTML URL-encoding Reference. Parameters are case-sensitive.

You can customize and localize user-facing templates.

/idp/startSSO.ping

This is the path used to initiate an unsolicited IdP-initiated SSO transaction during which a SAML response containing an assertion is sent to a service provider (SP). Typically, a systems integrator or developer creates one or more links to this endpoint in the IdP application or portal to allow users to initiate SSO to various SPs.

For information about allowing applications to retrieve configuration data from the PingFederate server over SOAP, see Web service interfaces and APIs.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`PartnerSpId` or `PARTNER`	The federation ID of the SP to whom the SAML response containing an assertion should be issued. This ID value is case-sensitive. One of these parameters is required unless the federation ID can be derived from `TargetResource` or `TARGET`.
`TargetResource` or `TARGET` (optional)	For SAML 2.0, the value of either parameter is passed to the SP as the `RelayState` element of a SAML response message. This is the PingFederate implementation of the SAML 2.0 indicator for a desired resource at the SP during IdP-initiated SSO. For SAML 1.x, the value is sent to the SP as a parameter named `TARGET`.The parameter value must be URL-encoded.
`InErrorResource` (optional)	Indicates where the user is redirected after an unsuccessful SSO. If this parameter is not included in the request, PingFederate redirects the user to the SSO error landing page hosted within PingFederate.

Parameter	Description
`Binding` (optional)	Indicates the binding to be used; allowed values are URIs defined in the SAML specifications. For example, the SAML 2.0 applicable URIs are: • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact` • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Post` When the parameter is not used, the default ACS URL configured for the SP-partner connection is used unless an ACS index is specified using the `ACSIdx` parameter.
`ACSIdx` (optional - SAML 2.0)	Specifies the index number of partner's ACS. For more information, see Setting Assertion Consumer Service URLs (SAML). Takes precedence over the `Binding` parameter if both are specified. If neither the binding nor index is specified in the call, the default ACS is used.
`IdpAdapterId` (optional)	Allows an application to call out what IdP adapter to use for authentication in a configuration with multiple IdP adapters. ⓘ **Note** This parameter might be overridden by policy based on authentication policies. For example, a CIDR Authentication Selector instance could enforce the use of a given adapter instance based on whether a user is on or off the network. For more information, see Authentication policies.
`ChangePassword`	If a request includes this parameter with a value of `true` and invokes an HTML Form Adapter instance, the user is redirected to the Change Password template and prompted to update the network password. ⓘ **Note** To use this parameter, the **Allow Password Changes** check box must be selected in the adapter configuration of the invoked HTML Form Adapter instance. For more information, see Configuring an HTML Form Adapter instance.
`RequestedFormat` (optional - SAML 2.0)	Allows control over the `NameId` format.
`vsid` (optional)	Specifies the virtual server ID. When absent, PingFederate uses the default virtual server ID, if specified, for the connection or the SAML federation ID defined in Server Settings. For more information, see Identifying the SP and Specifying federation information.
`PolicyAction` (optional)	The HTML Form Adapter immediately returns the value of this parameter in the `policy.action` attribute, allowing the policy to bypass the adapter in favor of an alternative authentication source, provided a rule matching the action is configured. When this parameter is set to `identity.registration` and the adapter is followed by a local identity profile, the user is directed to the registration page for the profile.

/idp/startSLO.ping

This is the path used to start an IdP-initiated SLO (under SAML 2.0) or an OpenID Connect (OIDC) logout. For more information see Asynchronous Front-Channel Logout. Typically, a systems integrator or developer creates one or more links to this endpoint in the protected resources of their IdP application or portal to allow users to end their sessions at various SPs. This endpoint uses the local PingFederate session to determine which SPs have been issued an SSO assertion and sends them a SAML logout request.

PingFederate sends SLO requests in the following sequence, which prioritizes synchronous logouts:

1. IdP adapters
2. SAML IdP partners
3. SP adapters
4. SAML SP partners
5. WS-Fed and OIDC partners in parallel

The `LogoutType` parameter lets you customize the SLO process. For example, you can change the SLO process so that it does not prioritize synchronous logouts.

To start with asynchronous logouts, you could chain the logouts. Start with `/idp/startSLO.ping?LogoutType=AsyncOnly`. Set the `TargetResource` to `/idp/startSLO.ping?LogoutType=SyncOnly`. In this case, you would also need to add the `/idp/startSLO.ping` endpoint as an allowed redirect for SLO on the Redirect Validation window.

To perform asynchronous and synchronous logouts in parallel, you could create a page with two iframes. One iframe points to `/idp/startSLO.ping?LogoutType=AsyncOnly` and the other points to `/idp/startSLO.ping?LogoutType=SyncOnly`.

The following table describes the HTTP parameters for this endpoint.

Parameter	Description
`TargetResource` (optional)	Indicates where the user is redirected after a successful SLO. If this parameter is not included in the request, PingFederate uses as a default the URL for a successful SLO as entered on the IdP Default URL window. The parameter value must be URL-encoded.
`InErrorResource` (optional)	Indicates where the user is redirected after an unsuccessful SLO. If this parameter is not included in the request, PingFederate redirects the user to the SLO error landing page hosted within PingFederate.
`Binding` (optional - SAML 2.0)	Indicates the binding to use. The allowed values are URIs defined in the SAML specifications. The SAML 2.0 applicable URIs are: • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact` • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST` • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect` • `urn:oasis:names:tc:SAML:2.0:bindings:SOAP` When the parameter is not used, the first SLO Service URL configured for the SP-partner connection is used. For more information, see Specifying SLO service URLs (SAML 2.0).

Parameter	Description
`LogoutType` (optional)	Controls which kinds of adapters and partners PingFederate will send SLO requests to. The allowed values are: • `All` (the default value) logs out all adapters and partners in the default sequence • `SyncOnly` logs out only SAML partners and adapters in series • `AsyncOnly` logs out only WS-Fed and OIDC partners in parallel ⓘ **Note** When WS-Fed partner logout entries exist, `LogoutType=AsyncOnly` might still cause the log out of all partners and adapters. The reason is that WS-Fed logout requests to a partner might reflect back to PingFederate. For example, a wsignout1.0 request to an IdP reflects back to PingFederate as a wsignoutcleanup1.0 request. When PingFederate receives the signout request, there is no context connecting it to the original request. As a result, PingFederate performs a full SLO.

/idp/writecdc.ping

This endpoint is used for SAML 2.0 IdP Discovery. This is the path used when the IdP wants to write to the common domain cookie (CDC) held within the user's browser. The information written to the cookie indicates from which IdP this user has authenticated.

The following table shows the one HTTP query parameter for this endpoint.

Parameter	Description
`TargetResource` (optional)	Indicates where the user is redirected after successful IdP Discovery. If this parameter is not included in the request, PingFederate redirects the user to the referrer in the HTTP header. If there is no `TargetResource` or referrer, the call to this endpoint will fail. The parameter value must be URL-encoded.

/pf/idprofile.ping

This endpoint is used for profile management. When profile management is enabled for customer identities, authenticated users can review and modify the local identity fields that have been configured to be shown on the profile management page, connect or disconnect third-party identity providers, also known as Social Connections to the end users on the profile management page, and delete their local identities if the option to do so has been enabled. Each local identity profile has its own profile management URL.

The following table shows the one HTTP query parameter for this endpoint.

Parameter	Description
`LocalIdentityProfileID`	Indicates which profile management page that PingFederate should serve to the authenticated users based on the ID of the local identity profile. **Tip** You can copy the profile management URL for a given local identity profile on its configuration summary window.

/pf/id/verification.ping

This endpoint is used for email ownership verification. When email ownership verification is enabled for customer identities, authenticated users can request additional email-verification notifications by accessing this endpoint.

The following table shows the one HTTP query parameter for this endpoint.

Parameter	Description
`LocalIdentityProfileID`	Indicates the local identity profile from which the authenticated users, who are requesting additional email-verification notifications, originate. **Tip** You can copy the email ownership verification endpoint for a given local identity profile on its configuration summary window.

/ext/pwdchange/Identify

The Change Password endpoint allows users to change their password through an HTML Form Adapter instance without submitting SSO requests. This endpoint requires one parameter, `AdapterId`; the parameter value is the identifier of the HTML Form Adapter instance that has been configured with such capability. For example, if the fully-qualified name of your PingFederate environment and the adapter ID are www.example.com and HTMLFormSimplePCV respectively, the resulting URL is /https://www.example.com/ext/pwdchange/Identify?AdapterId=HTMLFormSimplePCV.

The following table shows the one additional HTTP query parameter for this endpoint.

Parameter	Description
`TargetResource`	Indicates the desired destination after users have successfully change their network password. **Note** When target resource validation is enabled for this endpoint, as indicated by the **SLO and Other** check box on the **Security** > **System Integration** > **Redirect Validation** > **Local Redirect Validation** tab, PingFederate honors the URL only if the parameter value satisfies the configured requirement on the **Local Redirect Validation** tab. If the validation fails, PingFederate displays the default success or error message. For more information, see Configuring redirect validation.

/ext/pwdreset/Identify

The Account Recovery endpoint allows users to reset their password or unlock their account through an HTML Form Adapter instance without submitting SSO requests. This endpoint requires one parameter, `AdapterId`; the parameter value is the identifier of the HTML Form Adapter instance that has been configured with such capabilities. For example, if the fully-qualified name of your PingFederate environment and the adapter ID is www.example.com and HTMLFormSimplePCV respectively, the resulting URL is https://www.example.com/ext/pwdreset/Identify?AdapterId=HTMLFormSimplePCV.

The following table shows the one additional HTTP query parameter for this endpoint.

Parameter	Description
`TargetResource`	Indicates the desired destination after users have successfully reset their network password or unlocked their account. **Note** When target resource validation is enabled for this endpoint, as indicated by the **SLO and Other** check box on the **Security > System Integration > Redirect Validation > Local Redirect Validation** tab, PingFederate honors the URL only if the parameter value satisfies the configured requirement on the **Local Redirect Validation** tab. If the validation fails, PingFederate displays the default success or error message. For more information, see Configuring redirect validation.

/ext/idrecovery/Recover

The Username Recovery endpoint allows users to recover their username through an HTML Form Adapter instance without submitting SSO requests. This endpoint requires one parameter, `AdapterId`; the parameter value is the identifier of the HTML Form Adapter instance that has been configured with such capabilities. For example, if the fully-qualified name of your PingFederate environment and the adapter ID is www.example.com and HTMLFormSimplePCV respectively, the resulting URL is /https://www.example.com/ext/idrecovery/Recover?AdapterId=HTMLFormSimplePCV.

The following table shows the one additional HTTP query parameter for this endpoint.

Parameter	Description
`TargetResource`	Indicates the desired destination after users have successfully recovered their username. **Note** When target resource validation is enabled for this endpoint, as indicated by the **SLO and Other** check box on the **Security > System Integration > Redirect Validation > Local Redirect Validation** tab, PingFederate honors the URL only if the parameter value satisfies the configured requirement on the **Local Redirect Validation** tab. If the validation fails, PingFederate displays the default success or error message. For more information, see Configuring redirect validation.

Related links

- Customizable user-facing pages

- Localizing messages for end users

SP endpoints

PingFederate provides configuration options for a variety of endpoints.

The following sections describe the PingFederate service provider (SP) endpoints for SP services and system for cross-domain identity management (SCIM) inbound provisioning.

SP services

The following sections describe PingFederate service provider (SP) endpoints, including the query parameters that each accepts or requires. These endpoints accept either the HTTP GET or POST methods.

Begin each URL with the fully-qualified server name and port number of your PingFederate SP server; for example, https://www.example.com:9031/sp/startSSO.ping.

> **Important**
>
> When using the parameters `TargetResource` or `TARGET` with their own query parameters included, the parameter value must be URL-encoded. Any other parameters that contain restricted characters, such as many SAML URNs, also must be URL-encoded. For information about URL encoding, see third party resources such as HTML URL-encoding Reference. Parameters are case-sensitive.

/sp/startSSO.ping

This is the path used to initiate SP-initiated single sign-on (SSO). In this scenario, the SP issues an SSO request to the identity provider (IdP) asking for an SSO authentication response. Typically, a systems integrator or developer creates links to this endpoint in SP applications to allow users to access various protected resources through SSO using the IdP as an authentication authority.

For information about allowing applications to retrieve configuration data from the PingFederate server over SOAP, see Web service interfaces and APIs.

The following table shows the HTTP parameters for this endpoint.

> **Note**
>
> Some parameters described below can have multiple values. Specify these values by using multiple independent query string parameters of the same name.

Parameter	Description
`PartnerIdpId`	The federation ID of the IdP that authenticates the user and issues an assertion. This ID is case-sensitive. Required if more than one IdP connection is configured and `Domain` is not used, and SP authentication policies are turned off. Not required if SP authentication policies are turned on.

Parameter	Description
`SpSessionAuthnAdapterId`	The explicit SP adapter instance ID indicating the adapter to use to create an authenticated session or security context. Optional if SP authentication policies are turned off. Required if SP authentication policies are turned on unless the PingFederate SP server can determine the applicable SP adapter instance based on the target URL mapping configuration and the `TargetResource` or `TARGET` value at runtime.
`TargetResource` or `TARGET`	This parameter indicates the target applications where a successful SSO redirects the end-user. The parameter value must be URL-encoded. When this parameter is not provided in the URL, you can specify a default target resource in the administrative console, either for all IdP connections, for individual connections, or both. For more information, see Configuring default URLs and Configuring default target URLs.
`InErrorResource` (optional)	This parameter indicates where an unsuccessful SSO redirects the end-user. If this parameter is not included in the request, PingFederate redirects the user to the single log-out (SLO) error landing page hosted within PingFederate. For more information, see Customizable user-facing pages.

`Binding`
(optional)
	Indicates the binding to use; allowed values are URIs defined in the SAML specifications. For example, the SAML 2.0 applicable URIs are
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact`
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
	When the parameter is not used for SAML 2.0, the first SSO Service URL configured for the IdP-partner connection is used. For more information, see Specifying SSO service URLs (SAML).

`AllowCreate` (optional - SAML 2.0)	Controls the value of the `AllowCreate` attribute of the `NameIDPolicy` element in the AuthnRequest. The default is `true`.
`AuthenticatingIdpId` (optional - SAML 2.0)	This parameter indicates the preferred IdP for authenticating the user through an IdP proxy, such as PingOne for Enterprise. The parameter specifies the value of the `ProviderID` attribute in the `Scoping/IDPList/IDPEntry` element in the AuthnRequest. For more information, see section 3.4.1.3.1 of the OASIS SAML document . You can specify multiple values to build a preferred list.
`ForceAuthn` (optional - SAML 2.0 or OpenID Connect)	For SAML 2.0, this parameter controls the attribute of the same name in the AuthnRequest. For OpenID Connect, a value of `true` sets the `prompt` parameter in the authentication request to `login`. For more information about the authentication request and its parameter, see the OpenID Connect specification. The default is `false`.

Parameter	Description
`IsPassive` (optional - SAML 2.0 or OpenID Connect)	For SAML 2.0, this parameter controls the attribute of the same name in the AuthnRequest. For OpenID Connect, a value of `true` sets the `prompt` parameter in the authentication request to `none`. The default is `false`.
`RequestedACSIdx` (optional - SAML 2.0)	The index number of your site's Assertion Consumer Service, where you want the assertion to be sent.
`RequestedAcsUrl` (optional - SAML 2.0)	The URL of your site's Assertion Consumer Service, where you want the assertion to be sent.
`RequestedAuthnCtx` (optional - SAML 2.0 or OpenID Connect)	For SAML 2.0, this parameter indicates the requested authentication context of the assertion; allowed values include URIs defined in the SAML specifications. For more information, see the OASIS SAML document saml-authn-context-2.0-os.pdf. For OpenID Connect, the specified value becomes the `acr_values` parameter value in the authentication request. You can specify multiple values to build a preferred list.
`RequestedAuthnDeclRef` (optional - SAML 2.0)	An alternative to `RequestedAuthnCtx` above, indicating the requested authentication context of the assertion by declaring any URI reference. For more information see section 2.7.2.2 of the OASIS SAML document. You can specify multiple values to build a preferred list.
`RequestedBinding` (optional - SAML 2.0)	Indicates the binding requested for the response containing the assertion; allowed values are URIs defined in the SAML specifications.
`RequestedFormat` (optional - SAML 2.0)	Specifies the value for the Format attribute in the `NameIDPolicy` element of the AuthnRequest. If not specified, the AuthnRequest does not include the attribute.
`RequestedSPNameQualifier` (optional - SAML 2.0)	Indicates that the IdP should return the given name qualifier as part of the assertion used primarily to identify SP affiliations. For more information, see SP affiliations.
`vsid` (optional)	Specify the virtual server ID. When absent, PingFederate uses the default virtual server ID (if specified) for the connection or the SAML federation ID defined in Server Settings. For more information, see Identifying the partner and Specifying federation information.
`PolicyAction` (optional)	The HTML Form Adapter immediately returns the value of this parameter in the `policy.action` attribute, allowing the policy to bypass the adapter in favor of an alternative authentication source, provided a rule matching the action is configured. When this parameter is set to `identity.registration` and the adapter is followed by a local identity profile, the user is directed to the registration page for the profile.

If `SpSessionAuthnAdapterId` specifies an adapter, then that adapter is used to create an authenticated session for SP-initiated SSO. If there is no `SpSessionAuthnAdapterId`, the ultimate destination of the user after SSO, either the `TargetResource` or the default SSO success URL, is used along with the mappings defined in the administrative console on the Map URLs to Adapter Instances window. For more information, see Configuring target URL mapping.

Note that adapter selection for SP-initiated SSO is similar to that for IdP-initiated SSO except that, because the adapter ID depends on the SAML deployment, PingFederate cannot expect it from an IdP. Therefore, it uses only the URL mapping for adapter selection for SSO.

/sp/startSLO.ping

This is the path used to initiate SP-initiated SLO. Typically, a systems integrator or developer creates one or more links to this endpoint in the protected resources of their SP application, which allows users to end a session by sending a logout request to the IdP that authenticated the session.

Note that the IdP might send additional logout request messages to other SPs when it receives a logout request from a PingFederate server acting as an SP.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`TargetResource` (optional)	Indicates where a successful SLO redirects the user. If the request does not include this parameter, PingFederate uses the URL for a successful SLO as a default, as entered on the SP Default URLs window. Note that the parameter value must be URL-encoded.

`Binding`
(optional - SAML 2.0)
	Indicates the binding to use; allowed values are URIs defined in the SAML specifications. The SAML 2.0 applicable URIs are
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact`
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`
	`urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect`
	`urn:oasis:names:tc:SAML:2.0:bindings:SOAP`
	When the parameter is not used, the first SLO Service URL configured for the IdP-partner connection is used. For more information, see Specifying SLO service URLs (SAML 2.0).

`InErrorResource` (optional)	Indicates where an unsuccessful SLO redirects the user. If the request does not include this parameter, PingFederate redirects the user to the SLO error landing page hosted within PingFederate. For more information, see Customizable user-facing pages.

/sp/defederate.ping

This path terminates an account link created during SSO. Account linking provides a means for subject identification on the SP side. On the SP side, only users create and terminate links. The link contains the name identifier from the IdP, the IdP's federation ID, the adapter instance ID, and the local user identifier.

There are no HTTP parameters for this endpoint.

You can unlink a user session only if it was established during SSO using an existing account link on the SP side. If more than one SP session was established through account linking on the same PingFederate session, this endpoint will terminate each of those links. A local logout is also performed for any link that is terminated.

/sp/cdcstartSSO.ping

This endpoint is used for IdP-Discovery implementations. For more information, see Standard IdP Discovery . This endpoint is similar to `/sp/startSSO.ping` and accepts the same parameters, with the exception of `PartnerIdpId` and `vsid`. Instead of this parameter, the server attempts to use the common domain cookie to determine the IdP.

/sp/startAttributeQuery.ping

This endpoint initiates an Attribute Query with a SAML 2.0 IdP. For more information, see Attribute Query and XASP.

The following table shows the HTTP parameters for this endpoint.

> **Note**
>
> Some parameters described below can have multiple values. Specify these values by using multiple independent query string parameters of the same name.

Parameter	Description
`Subject`	Uniquely identifies the user to the IdP. When user authenticates with an X.509 certificate, this is the Subject DN, which must be URL-encoded.
`Issuer` (optional)	The IssuerDN from the user's X.509 certificate, when X.509 attribute sharing profile (XASP) is used, which uniquely identifies the entity that issued the user's certificate. The parameter must be URL-encoded. **Note** When specified this parameter overrides the `Subject` parameter.
`PartnerIdpId` (except for XASP)	Used to identify the specific IdP partner to which the attribute query should be sent. Without this parameter, the Subject and Issuer are used to determine the correct IdP. **Note** For XASP, this parameter overrides both the `Subject` and `Issuer` parameters.
`Format` (required for XASP, otherwise optional)	Identifies the name-identifier format of the `Subject` query parameter. If included, the value must be one of the SAML 2.0 Name Identifier Format URIs. For more information, see section 8.3 of the SAML specifications. **Note** For XASP, this parameter must be set to `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName` If not specified, the parameter defaults to `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` . The parameter must be URL-encoded.

Parameter	Description
`AppId`	The unique identifier of the initiating application.
`SharedSecret`	Used to authenticate the initiating application. Both the `AppId` and `SharedSecret` values must match those defined on the Security > System Integration > Service Authentication window. **Important** To avoid recording this parameter in web server logs, only pass it in the message body using the HTTP POST method.
`RequestedAttrName` (optional)	A name of a user attribute requested from the IdP. For each desired user attribute, include this parameter. If this parameter is not present, then the IdP returns all allowable user attributes. You can specify multiple values to build a preferred list.
`vsid` (optional)	Specify the virtual server ID. When absent, PingFederate uses the default virtual server ID, if specified, for the connection or the SAML federation ID defined in Server Settings. For more information, see Identifying the partner and Specifying federation information.

SCIM inbound provisioning endpoints

PingFederate supports system for cross-domain identity management (SCIM) inbound provisioning and provides four endpoints.

The four endpoints are:

- `/pf-scim/v1/Users`
- `/pf-scim/v1/Groups`
- `/pf-scim/v1/Schemas`
- `/pf-scim/v1/ServiceProviderConfigs`

These endpoints are defined in the following SCIM 1.1 specifications:

- SCIM Core Schema
- SCIM Specification

Begin each endpoint with the fully-qualified server name and port number of your PingFederate server, for example: https://pingidentity.com:9031/pf-scim/v1/Users.

/pf-scim/v1/Users

The users endpoint is where client applications make HTTP requests to create, retrieve, update, and delete or deactivate users. This REST-based endpoint accepts POST, GET, PUT, and DELETE methods, as described in the following table.

Note

HTTP requests must be made using either Basic or client-certificate application authentication. JSON is currently the only supported format for the HTTP message body.

HTTP method	Description
POST	`/pf-scim/v1/Users` • Sends user attributes in JSON format—defined in the SCIM Core Schema—to create a new user. • A successful response is indicated by an HTTP 201 status code and a message body containing the user record that has been added to the target datastore. The user ID is set as the `id` attribute in the JSON response, and the full URL to reference the user is in the HTTP response Location header. For an existing user, you can also use the POST method to either update or delete or disable a user record by appending the user ID to the path in the format of `/pf-scim/v1/Users/user_id` and setting the request header `X-HTTP-Method-Override` value to PUT or DELETE, respectively. For more information, see the PUT and DELETE method descriptions at the end of this topic.

GET	`/pf-scim/v1/Users`
	• Retrieves all user records.
	• A successful response is indicated by an HTTP 200 status code and a list of all users and their attributes.
	`/pf-scim/v1/Users/user_id`
	• Retrieves the user record of a specific user.
	• A successful response is indicated by an HTTP 200 status code and the requested user record.
	`/pf-scim/v1/Users?attributes=attribute`
	• Retrieves the specific attribute from all users.
	• A successful response is indicated by an HTTP 200 status code and a list of the desired attribute from all users.
	Note
	For more information, see 3.2.2 List/Query Resources in SCIM Specification.
	`/pf-scim/v1/Users?filter=filter`
	• Retrieves resources based on the filter.
	• A successful response is indicated by an HTTP 200 status code and a list of resources matching the filter.
	Note
	For more information, see 3.2.2.1 Filtering in SCIM Specification.
	`/pf-scim/v1/Users?sortBy=attribute&sortOrder=ascending|descending`
	• Retrieves all user records and sorts them based on a specific attribute in ascending or descending order.
	• A successful response is indicated by an HTTP 200 status code and a sorted result set. Depending on the implementation of the target datastore, the target datastore might not return the user records that do not contain a value for that specific attribute as indicated by the `sortBy` parameter in the request.
	Note
	For more information, see 3.2.2.2 Sorting in SCIM Specification.
	`/pf-scim/v1/Users?startIndex=xy]`
	• Retrieves the user records starting with a specific index number, a positive integer *x*. If the optional `count` parameter is included, with a positive integer *y*, the endpoint limits the number of user records in the result set.
	• A successful response is indicated by an HTTP 200 status code and a limited set of user records.
	Note
	For more information, see 3.2.2.3 Pagination in SCIM Specification.

HTTP method	Description
	Tip You can use a combination of the aforementioned parameters in one query to narrow your search results.
PUT	`/pf-scim/v1/Users/user_id` • Updates user attributes for the specified user, using JSON in the body of the HTTP request. Attributes not included in the request are set to a default value in the datastore. • A successful PUT operation returns an HTTP 200 status code and the entire updated user record within the response body.
DELETE	`/pf-scim/v1/Users/user_id` • Deletes or disables the user record for the specified user. Note that whether a user is deleted or disabled is determined by the selection of the SCIM DELETE message behavior option on the Delete/Disable Users tab in the applicable identity provider (IdP) connection. • A successful response is indicated by an HTTP 200 status code.

Note

For a list of HTTP error codes that might be returned, see 3.9 HTTP Response Codes in SCIM Specification.

/pf-scim/v1/Groups

The groups endpoint is where client applications make HTTP requests to create, retrieve, update, and delete groups.

Note

Inbound provisioning for groups is a per-connection, optional feature. To enable group provisioning, select the **User and Group Support** option on the **Connection Type** tab when configuring the applicable IdP connection.

This REST-based endpoint accepts POST, GET, PUT, and DELETE methods, as described in the following table.

Note

HTTP requests must be made using either Basic or client-certificate application authentication. JSON is currently the only supported format for the HTTP message body.

HTTP method	Description
POST	`/pf-scim/v1/Groups` - Sends group attributes in JSON format—defined in the SCIM Core Schema—to create a new group. - A successful response is indicated by an HTTP 201 status code and a message body containing the group record that has been added to the target datastore. The group ID is set as the `id` attribute in the JSON response, and the full URL to reference the group is in the HTTP response Location header. For an existing group, you can also use the POST method to either update or delete the group by appending the group ID to the path, in the format of `/pf-scim/v1/Groups/group_id`, and setting the request header `X-HTTP-Method-Override` value to PUT or DELETE, respectively. For more information, see the PUT and DELETE method descriptions at the end of this topic.

GET	`/pf-scim/v1/Groups`
	• Retrieves all group records.
	• A successful response is indicated by an HTTP 200 status code and a list of all groups and their attributes.
	`/pf-scim/v1/Groups/group_id`
	• Retrieve the group record of a specific group.
	• A successful response is indicated by an HTTP 200 status code and the requested group record.
	`/pf-scim/v1/Groups?attributes=attribute`
	• Retrieves the specific attribute from all groups.
	• A successful response is indicated by an HTTP 200 status code and a list of the desired attribute from all groups.
	Note
	For more information, see 3.2.2 List/Query Resources in SCIM Specification.
	`/pf-scim/v1/Groups?filter=filter`
	• Retrieves resources based on the filter.
	• A successful response is indicated by an HTTP 200 status code and a list of resources matching the filter.
	Note
	For more information, see 3.2.2.1 Filtering in SCIM Specification.
	`/pf-scim/v1/Groups?sortBy=attribute&sortOrder=ascending|descending`
	• Retrieves all group records and sorts them based on a specific attribute in ascending or descending order.
	• A successful response is indicated by an HTTP 200 status code and a sorted result set. Depending on the implementation of the target datastore, the target datastore might not return the group records that do not contain a value for that specific attribute as indicated by the `sortBy` parameter in the request.
	Note
	For more information, see 3.2.2.2 Sorting in SCIM Specification.
	`/pf-scim/v1/Groups?startIndex=xy]`
	• Retrieves the group records starting with a specific index number, a positive integer *x*. If the optional `count` parameter is included, with a positive integer *y*, the endpoint limits the number of user records in the result set.
	• A successful response is indicated by an HTTP 200 status code and a limited set of group records.
	Note
	For more information, see 3.2.2.3 Pagination in SCIM Specification.

HTTP method	Description
	Tip You can use a combination of the aforementioned parameters in one query to narrow your search results.
PUT	`/pf-scim/v1/Groups/group_id` • Updates group attributes for the specified group, using JSON in the body of the HTTP request. Attributes not included in the request are set to a default value in the datastore. • A successful PUT operation returns an HTTP 200 status code and the entire updated group record within the response body.
DELETE	`/pf-scim/v1/Groups/group_id` • Deletes the group record for the specified group. • A successful response is indicated by an HTTP 200 status code.

Note

For a list of HTTP error codes that might be returned, see 3.9 HTTP Response Codes in SCIM Specification.

/pf-scim/v1/Schemas

The schemas endpoint is where a client can retrieve a resource's schema. This REST-based endpoint accepts GET method as described in the following table.

Note

HTTP requests must be made using either Basic or client-certificate application authentication. JSON is currently the only supported format for the HTTP message body.

HTTP method	Description
GET	Retrieves the resource's schema for an IdP connection based on the authentication information. A successful response is indicated by an HTTP 200 status code and the results in the message body.

Example

```
$ curl -u basicUser 'https://localhost:9031/pf-scim/v1/Schemas' | python -m json.tool
{
    "attributes": [
        {
            "caseExact": false,
            "description": "Unique identifier for the SCIM resource as defined by the Service Provider. Each
representation of the resource MUST include a non-empty id value. This identifier MUST be unique across the Service
Provider's entire set of resources. It MUST be a stable, non-reassignable identifier that does not change when the
same resource is returned in subsequent requests. The value of the id attribute is always issued by the Service
Provider and MUST never be specified by the Service Consumer. REQUIRED.",
            "multiValued": false,
            "name": "id",
            "readOnly": true,
            "required": true,
            "schema": "urn:scim:schemas:core:1.0",
            "type": "string"
        },
        ...
    ],
    "description": "Core User",
    "endpoint": "/Users",
    "id": "urn:scim:schemas:core:1.0:User",
    "name": "User",
    "schema": "urn:scim:schemas:core:1.0"
}
```

/pf-scim/v1/ServiceProviderConfigs

This service provider (SP) configuration endpoint is where developers can retrieve detailed information on the PingFederate SCIM 1.1 implementation. When you enable inbound provisioning for an SP PingFederate server, an HTTP GET request to this endpoint returns a JSON response outlining SCIM 1.1 compliance details.

> **Note**
>
> The `/pf-scim/v1/ServiceProviderConfigs` endpoint does not require authentication. JSON is currently the only supported format for the HTTP message body.

Example

```
$ curl https://localhost:9031/pf-scim/v1/ServiceProviderConfigs
{
  "schemas": ["urn:scim:schemas:core:1.0"],
  ...
  "patch": {
    "supported":false
  },
  "bulk": {
    "supported":false
  },
  "filter": {
    "supported":true
  },
  "changePassword" : {
    "supported":true
  },
  "sort": {
    "supported":false
  },
  "etag": {
    "supported":false
  },
  "xmlDataFormat": {
    "supported":false
  },
  "authenticationSchemes": [
    {
      "name": "HTTP Basic",
      "description": "Authentication using HTTP Basic",
      ...
      "type":"httpbasic"
    },
    {
      "name": "TLS Client Certificate",
      "description": "Authentication via TLS Client Certificate",
      ...
      "type":"tls"
    }
  ]
}
```

System-services endpoints

System-services endpoints generally apply to the PingFederate server, whether used as an identity provider (IdP), service provider (SP), or both. Parameters are case-sensitive.

/pf/heartbeat.ping

This endpoint returns an HTTP status code of 200 and a message body of OK if the PingFederate runtime server is up and functional. You can customize the message by modifying a PingFederate property and a Velocity template file. For more information, see Customizing the heartbeat message.

ⓘ **Note**

If a GET request receives a connection error or an HTTP status code other than 200, the server associated with the endpoint is down or malfunctioning.

Load balancers can use this endpoint to determine the status of PingFederate independently of checks used to determine the status of the supporting hardware.

You can also configure the server to provide regular status information to a network-management utility. For more information, see Runtime monitoring using JMX.

/pf/adapter2adapter.ping

This endpoint initiates direct IdP-to-SP adapter mapping, when that feature is configured in the Adapter-to-Adapter Mappings window. For more information, see Adapter-to-adapter mappings.

ⓘ **Note**

To prevent users from circumventing the SP authentication policies, this endpoint becomes inactive when SP authentication policies are enabled but IdP authentication policies are disabled. Administrators can configure SP authentication policies for the internal users to re-enable access to protected resources.
For information, see Configuring SP authentication policies for internal users.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`TargetResource` (optional)	Indicates where the user is redirected after a successful SSO. If this parameter is not included in the request, PingFederate redirects the user to a default location if one is specified in the Applications > Integration > SP Default URLs window.
`InErrorResource` (optional)	Indicates where the user is redirected if the SSO is unsuccessful. If this parameter is not included in the request, PingFederate redirects the user to the SSO error landing page hosted within PingFederate. For more information, see Customizable user-facing pages.
`IdpAdapterId` (optional)	Indicates the IdP adapter instance to use for authentication if more than one IdP adapter is configured in adapter-to-adapter mappings.
`SpSessionAuthnAdapterId` (optional)	Indicates the SP adapter instance to be used. If not provided and more than one SP adapter instance is configured with adapter-to-adapter mapping, PingFederate selects one based on entries defined in the Applications > Integration > Target URL Mapping window. For more information, see Configuring target URL mapping.

Parameter	Description
`ChangePassword`	If a request includes this parameter with a value of `true` and invokes an HTML Form Adapter instance, the user is redirected to the Change Password template and prompted to update the network password. **Note** To use this parameter, the **Allow Password Changes** check box must be selected in the adapter configuration of the invoked HTML Form Adapter instance. For more information, see Configuring an HTML Form Adapter instance.
`PolicyAction` (optional)	The HTML Form Adapter immediately returns the value of this parameter in the `policy.action` attribute, allowing the policy to bypass the adapter in favor of an alternative authentication source, provided a rule matching the action is configured. When this parameter is set to `identity.registration` and the adapter is followed by a local identity profile, the user is directed to the registration page for the profile.

/pf/sts.wst

This endpoint initiates direct security token service (STS) token-to-token exchange and token validation from an IdP token processor to an SP token generator, when that feature is configured in the Token Translator Mappings window. For more information, see Token translator mappings.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`TokenProcessorId`	Indicates the IdP token processor to use in the mapping. Required when multiple IdP token processors are configured in token-to-token mappings.
`TokenGeneratorId`	Indicates the SP token generator to use in the mapping. Required when multiple SP token generators are configured in token-to-token mappings.

Important

If mutual SSL/TLS is used for authentication, you must configure a secondary PingFederate listening port used by partners or STS clients for the relevant endpoints— `.ssaml` and `.wst`. For more information, see Configuring PingFederate properties.

/pf/sts_mex.ping

This endpoint returns STS metadata for use in expediting configuration of web-service applications.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`PartnerSpId`	The connection ID of the SP to whom the SAML token will be issued. This parameter determines the connection for which metadata will be generated.
`PartnerIdpId`	The connection ID of the IdP issuing the SAML token to be consumed by PingFederate. This parameter determines the connection for which the metadata will be generated.
`vsid` (optional)	Specify the virtual server ID. If absent, PingFederate uses the default virtual server ID (if specified) for the connection or the federation ID defined on the System > Server > Protocol Settings > Federation Info tab.

> **Note**
>
> If your partner fails to retrieve metadata when sending both the `PartnerSpId` or the `PartnerIdpId`, and the `vsid` query parameters, perhaps it is only capable of sending one query parameter in such requests. An alternative metadata exchange endpoint that includes the virtual server ID information should resolve the issue.
> For more information, see Constructing an alternative metadata exchange endpoint.

/pf/federation_metadata.ping

This endpoint returns SAML and WS-Federation metadata.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`PartnerSpId`	The connection ID of the SP to whom the assertions or tokens are issued. This parameter determines the connection for which metadata is generated.
`PartnerIdpId`	The connection ID of the IdP issuing the assertions or tokens to be consumed by PingFederate. This parameter determines the connection for which the metadata is generated.
`vsid` (optional)	Specify the virtual server ID. If absent, PingFederate generates the metadata based on the connection's default virtual server ID, if two or more virtual server IDs are defined, or the federation ID defined on the System > Server > Protocol Settings > Federation Info tab.

> **Note**
>
> If your partner fails to retrieve metadata when sending both the `PartnerSpId` or the `PartnerIdpId`, and the `vsid` query parameters, perhaps it is only capable of sending one query parameter in such requests. An alternative metadata exchange endpoint that includes the virtual server ID information should resolve the issue.
> For more information, see Constructing an alternative metadata exchange endpoint.

Constructing an alternative metadata exchange endpoint

You can embed virtual server ID information into a security token service (STS) metadata exchange endpoint or a SAML and WS-Federation metadata exchange endpoint.

About this task

This process is useful for scenarios where partners prefer to retrieve metadata by sending one query parameter such as `PartnerSpId` or `PartnerIdpId`, instead of two query parameters such as `PartnerSpId` or `PartnerIdpId` and `vsid`.

Steps

1. Construct a JSON object containing a key-value pair of the virtual server ID by using the following format. `{"vsid":"<VirtualServerIdValue>"}`

   Example:

   For example, if the virtual server ID is `Engineering`, the JSON object is `{"vsid":"Engineering"}`.

2. Base64url-encode the JSON object.

   Example:

   For example, if the JSON object is `{"vsid":"Engineering"}`, the base64url-encoded value is `eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ`.

   For more information about base64url, see tools.ietf.org/html/rfc4648.

3. Insert the base64url-encoded value prefixed with a forward slash into the metadata exchange endpoints, described as follows:

   Federation metadata endpoint (`/pf/federation_metadata.ping`)

   Between `/pf` and `/federation_metadata.ping`.

   STS metadata endpoint (`/pf/sts_mex.ping`)

   Between `/pf` and `/sts_mex.ping`.

   Example:

   For example, if the base64url-encoded value is `eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ`, the metadata exchange endpoints embedding with the virtual server ID are:

   Federation metadata endpoint

   `/pf/eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ/federation_metadata.ping`

   Example: https://idp.example.com:9031/pf/eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ/federation_metadata.ping?PartnerSpId=sp.example.org

   STS metadata endpoint

   `/pf/eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ/sts_mex.ping`

Example: https://idp.example.com:9031/pf/eyJ2c2lkIjoiRW5naW5lZXJpbmcifQ/sts_mex.ping?PartnerSpId=sp.example.org

Authentication API

The PingFederate authentication application programming interface (API) is a JSON-based API that enables end-user interactions, such as credential prompts, to be handled by an external web application. This API does so by providing access to the current state of the flow as an end user steps through a PingFederate authentication policy.

Authentication flows are initiated through browser-based single sign-on (SSO) application endpoints, such as `/idp/startSSO.ping`, or a protocol request, such as an OpenID Connect (OIDC) authentication request received at the authorization endpoint: `/as/authorization.oauth2`. As PingFederate runs the configured authentication policy, if it encounters an API-capable adapter or selector, and an authentication application is configured for the policy, PingFederate redirects to the authentication application's URL, passing the ID of the flow in the `flowId` query parameter.

The authentication application can then retrieve the current state of the flow by issuing a **`GET`** request to the `/pf-ws/authn/flows/{flowId}` endpoint. The `_links` field in the response lists the available authentication actions that can be performed from the current state. To invoke an action, the authentication application sends a **`POST`** request to the `/pf-ws/authn/flows/{flowId}` endpoint.

The API client can choose authentication actions using either an `action` query parameter or a custom content type. The `action` query parameter should be used when your networks block custom content types, and is specified using the following format:

```
/pf-ws/authn/flows/<flowId>?action=<actionID>
```

For example, to check username and password, pass the `checkUsernamePassword` action ID through the query parameter as follows:

```
/pf-ws/authn/flows/<flowId>?action=checkUsernamePassword
```

For the custom content type, the action ID is specified using the Content-Type HTTP request header in the following format:

```
application/vnd.pingidentity.<actionId>+json
```

For example, to indicate a username and password check from the HTML Form Adapter, specify the following:

```
application/vnd.pingidentity.checkUsernamePassword+json
```

When the application invokes an action, PingFederate responds either with the next state for the flow or an error.

When the user completes the authentication policy steps successfully, the authentication API returns a `RESUME` status to the authentication application. This status indicates that the API client should redirect the user's browser to the `resumeUrl` specified in the response. PingFederate will then be responsible for the final step in the flow, such as passing a SAML assertion to a partner. A `RESUME` status will also be sent if PingFederate encounters an identity provider (IdP) connection in the policy tree, or an IdP adapter or selector that is not API-capable. When the API client redirects the user, PingFederate will take the steps needed to invoke the authentication source.

If the user has interacted with an authentication application and the flow terminates with an error, the API client will receive a `FAILED` status from the API.

> **Note**
>
> To avoid issues with third-party cookies in some browsers, you should give the authentication application the same parent domain as the PingFederate authentication API URL that the application accesses. This could be a common domain of PingFederate's base URL or one of PingFederate's defined virtual hosts.

Key concepts

Flow

The SSO transaction invoking the authentication API.

States

The available states (if any) for a given API-enabled adapter or selector.

Current state

Indicates what the adapter or selector is ready to do next.

Actions

The available actions (if any) for a given state.

Session management

The authentication API endpoint has the same domain as PingFederate's other application endpoints and shares the PingFederate session cookies with those endpoints. This ensures that session data created by API applications can be retrieved when the user interacts with PingFederate's other endpoints and vice versa.

> **Note**
>
> Because the authentication API relies on the PingFederate session cookies, only browser-based web applications can currently make use of the API. Server-side web applications are not supported.

In order for PingFederate to manage session cookies correctly, the Javascript-written authentication application must set the `withCredentials` flag on the XMLHttpRequest object to `true`.

To protect against cross-site request forgeries, API clients are also required to include an X-XSRF-Header HTTP request header with each request, for example: `X-XSRF-Header: PingFederate`. This custom header ensures that browsers enforce cross-origin resource sharing (CORS) policies when API requests are sent. This header can have any value.

Authentication API Explorer

PingFederate includes an API Explorer, which allows you to view the states, actions, and models available for the various API-capable adapters and selectors included in your PingFederate environment. The endpoint for the Authentication API Explorer is `/pf-ws/authn/explorer`.

You can download a Postman collection file from Authentication > Integration > Authentication API Applications. The file contains information converted from the Explorer into a Postman collection. You can then import the collection file into the Postman application. The collection file provides every possible action of every state in every plugin deployed in the PingFederate instance. It contains:

- All API-capable plugins, both adapters and selectors, with all of their states and actions
- Pre-populated body containing information about required information types and parameters
- Headers for API calls, both X-XSRF-Header and Content-Type

For more information, see Exploring the authentication API.

JavaScript widget for the PingFederate authentication API

The JavaScript Widget for the PingFederate authentication API is a customizable JavaScript library that provides the capabilities of the HTML Form Adapter and Identifier First Adapter through the authentication API:

- User Login
- Trouble Signing in
- Trouble with Username
- Password Reset

The widget is a ready-to-use drop-in bundle with CSS and customizable templates. This alternative to the PingFederate templates provides a sign-in experience as a single page application. For more information, go to JavaScript Widget for the PingFederate Authentication API.

Exploring the authentication API

The Authentication API Applications window includes an Authentication API Explorer that lets you view the states, actions, and models available for the various API-capable adapters and selectors included in your PingFederate environment.

Steps

1. Enable the authentication API and Authentication API Explorer.
    1. Go to Authentication > Integration > Authentication API Applications.
    2. Select the Enable Authentication API check box.
    3. Select the Enable API Explorer check box.
2. Configure an authentication application for the Authentication API Explorer.
    1. Go to Authentication > Integration > Authentication API Applications.

2. Click Add Authentication Application.

3. In the Authentication Application window, configure each field as described in the following table.

Field	Description
Name	A name of the authentication application, such as `Authentication API Explorer`.
Description	An optional description of the authentication application, such as `Explore the authentication API!`
URL	A combination of the PingFederate base URL and the application path `/pf-ws/authn/explorer`. For example, if the base URL is https://localhost:9031, enter: `https://localhost:9031/pf-ws/authn/explorer`. You can find your PingFederate base URL at System > Server > Protocol Settings > Federation Info.
Additional Allowed Origins	Any additional allowed origins. For more information, see Configuring authentication applications. If you are using a PingFederate base URL of https://localhost:9031 for testing purposes, you can skip this field.

4. In the Authentication Application window, click Save.

5. In the Authentication API Applications window, click Save.

3. In the Authentication API Explorer, view the available states, actions, and models for any API-capable adapter or selector in your PingFederate environment:

   1. Go to the URL of the Authentication API Explorer, such as https://localhost:9031/pf-ws/authn/explorer

   2. In the Authentication Adapter/Selector list, select an authentication adapter or selector.

      Result:

   The Authentication API Explorer displays a list of states. You can then inspect the following items for any given state:

   - The state purpose
   - The state data model (if any)
   - The available action of actions (if any)
   - The action data model (if any) for a given action
   - The errors (if any) for a given action

   The developers of your web applications use this information to create the desired authentication experience.

4. Explore the authentication API through a request without an authentication policy:

   1. Configure a use case to use an instance of the HTML Form Adapter for authentication.

For example, you can create an service provider (SP) connection that uses an instance of the HTML Form Adapter for authentication. For more information, see Configuring a sample use case.

2. Go to Authentication > Integration > Authentication API Applications.

3. In the Default Authentication Application list, select the authentication application that represents the Authentication API Explorer. Click Save.

4. Initiate a request supported by the use case.

   Result:

   When PingFederate receives your request, it determines from your use case that it should invoke the HTML Form Adapter. Because the HTML Form Adapter is API-enabled and you have configured the Authentication API Explorer to be the default authentication application, instead of returning the Sign On page (from the HTML Form Adapter), PingFederate redirects the browser to the Authentication API Explorer with a `flowid` query parameter, such as https://localhost:9031/pf-ws/authn/explorer?flowId=Tt9n7.

   The Authentication API Explorer opens and pre-populates the Flow ID field with the flow ID value generated by PingFederate.

5. In the Authentication API Explorer, click Get next to the pre-populated flow ID value.

   Result:

   The Authentication API Explorer displays a JSON response as the result of the GET request. This response contains information that the web application requires to proceed further. For instance, the `status` parameter value indicates the current state of the request. Because the sample request invokes the HTML Form Adapter, the current state should be `USERNAME_PASSWORD_REQUIRED`.

   At the end of the result is a hyperlink to the current state. In this example, when you select the current state link, the Authentication API Explorer jumps to the USERNAME_PASSWORD_REQUIRED state and expands its contents for further review.

From this point, you can review the state data model and move the request further by selecting the appropriate action and action data, if it's required.

5. Explore the authentication API through a request with an authentication policy:

   1. Go to Authentication > Integration > Authentication API Applications.

   2. In the Default Authentication Application list, select Select.

      Result:

   No application is designated as the default authentication application.

   1. Define an authentication policy to use the Authentication API Explorer:

      1. Go to Authentication > Policies > Policies, and click Add Policy.

      2. Enter a policy name and optionally a description.

      3. In the Authentication Application list, select the authentication application that represents the Authentication API Explorer.

4. Select the HTML Form Adapter instance that has been mapped to the use case you configured in step 4a.

   For both the Fail and Success policy paths, select Done.

5. Click Done.

6. Select the IdP Authentication Policies check box.

7. Click Save.

2. Initiate a request supported by the use case you configured in step 4a.

3. Use the Authentication API Explorer to learn more about the authentication API.

6. Generate and configure a Postman collection JSON file.

   1. Go to the Authentication API Explorer.

   2. Click What is the PingFederate Authentication API? to expand the top panel

   3. Click the Postman Collection button.

      Result:

      The Authentication API Explorer downloads the Postman collection file.

   4. Open the Postman application, and import the file.

   5. In Postman, manually configure the following:

      - The `flowid` collection variable. When PingFederate receives a request, it generates a `flowid`, such as `Tt9n7`, and displays it in the Flow ID field on the Authentication API Explorer page.

      - The `baseUrl` collection variable. This is the base URL of your Authentication API Explorer, such as https://localhost:9031.

      - The PingFederate cookie.

        You must also modify the body, if one exists, to ensure that API calls work correctly.

Mobile application authentication through REST APIs

In PingFederate, you can configure mobile applications to authenticate through REST APIs as OAuth clients without needing to handle HTTP redirections. When authentication is complete, the applications receive an OAuth authorization code, or an access token and possibly an OpenID Connect (OIDC) ID token.

Single-page web applications can also use redirectless mode if administrators configure them in PingFederate as authentication applications. Learn more in Configuring authentication applications. Web-based authentication applications must be highly trusted.

Administrators can allow an OAuth client to initiate authorization directly through the authentication API:

1. Go to the client's configuration in the Client page.

2. Select the Allow Authentication API Redirectless Mode checkbox.

When enabling this feature, consider the following:

- Redirection URLs are optional, but without a redirection URL, browser-based OAuth flows do not work.
- This flow does not support the user-facing scope consent page, Request for Approval.

  > **Note**
  >
  > Enabling this feature automatically enables the **Bypass Authorization Approval** feature and **Restrict Common Scopes** feature.

- The application must manage the `PF` cookie by ensuring each request uses the cookie value from the previous server response. If persistent authentication sessions are configured, the same applies to the `PF.PERSISTENT` cookie.

To authenticate with this method, an OAuth client makes two or more API requests:

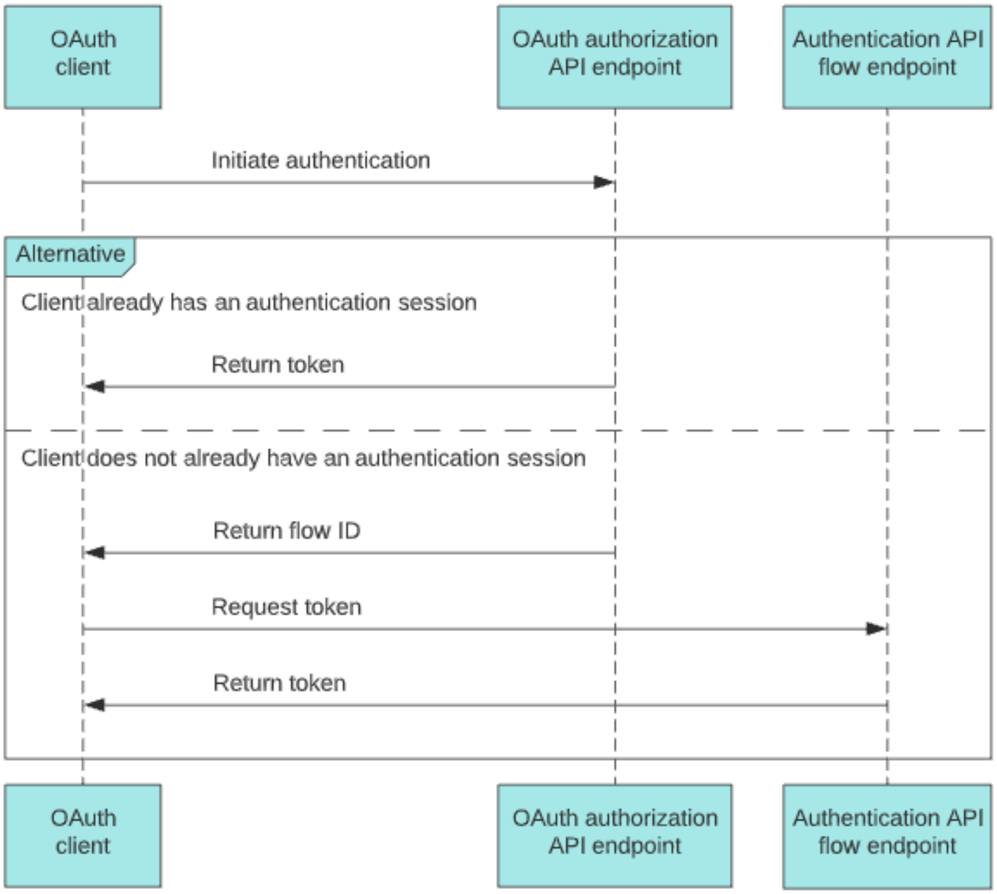
1. The OAuth client initiates authentication by calling the OAuth 2.0 authorization endpoint `/as/authorization.oauth2` to get a flow ID and other information it needs for the next request. The client must specify the pi.flow response mode in this call. See the first request sample below.

   > **Note**
   >
   > If a valid authentication session already exists when the OAuth client makes the first request, the client will receive a response with a token. In this case, the client does not need to make the next request.

2. The client calls the authentication API flow endpoint `/pf-ws/authn/flows/{flow_id}` to get the token. Learn more in the Second request sample below.

   > **Note**
   >
   > In some cases, depending on the configuration of the authentication policy, the client must make more than two requests to get a token.

Mobile application authentication through REST APIs


OAuth client
OAuth authorization API endpoint
Authentication API flow endpoint
Initiate authentication
Alternative
Client already has an authentication session
Return token
Client does not already have an authentication session
Return flow ID
Request token
Return token
OAuth client
OAuth authorization API endpoint
Authentication API flow endpoint

First request sample

```
GET /as/authorization.oauth2?client_id=im_client&response_type=token&response_mode=pi.flow HTTP/1.1
Host: www.example.com:9031
```

Sample response 1 to the first request

If a valid authentication session does not already exist, the OAuth client receives a response that provides the information the client needs for the next request (see the second request sample).

```
{
    "id": "PyH5g", // Flow ID
    "pluginTypeId": "7RmQNDWaOnBoudTufx2sEw",
    "status": "USERNAME_PASSWORD_REQUIRED",
    "showRememberMyUsername": false,
    "showThisIsMyDevice": false,
    "thisIsMyDeviceSelected": false,
    "showCaptcha": false,
    "rememberMyUsernameSelected": false,
    "_links": {
        "self": {
            "href": "https://www.example.com:9031/pf-ws/authn/flows/PyH5g"
        },
        "checkUsernamePassword": {
            "href": "https://www.example.com:9031/pf-ws/authn/flows/PyH5g"
        }
    }
}
```

The `self` link in the response shows that the second request must call the `pf-ws/authn/flows` endpoint instead of the authorization endpoint. The value of the `id` field is the ID of the flow that was created. The client must append the flow ID to the `pf-ws/authn/flows` endpoint for subsequent API requests.

Sample response 2 to the first request

If a valid authentication session already exists, the OAuth client receives a response with a token, so the client does not need to make the second request.

```
{
    "id": "PyH5g", // Flow ID
    "pluginTypeId": "7RmQNDWaOnBoudTufx2sEw",
    "status": "COMPLETED",
    "authorizeResponse": {
        "access_token": "000144Qlv9eqpBk03ngAd7M35Gaj41Mgisk",
        "token_type": "Bearer"
    "_links": {
        "self": {
            "href": "https://www.example.com:9031/pf-ws/authn/flows/PyH5g"
        }
    }
}
```

Second request sample

```
POST /pf-ws/authn/flows/PyH5g HTTP/1.1
Host: www.example.com:9031
X-XSRF-Header: PingFederate
Content-Type: application/vnd.pingidentity.checkUsernamePassword+json
Cookie: PF=8PgutwFizNS7EoaiB0qVsa

{
  "username": "joe",
  "password": "Password1"
}
```

Sample response to the second request

```
{
    "id": "PyH5g",
    "pluginTypeId": "7RmQNDWaOnBoudTufx2sEw",
    "status": "COMPLETED",
    "authorizeResponse": {
        "access_token": "000144Qlv9eqpBk03ngAd7M35Gaj41Mgisk",
        "token_type": "Bearer"
    "_links": {

        "self": {
            "href": "https://www.example.com:9031/pf-ws/authn/flows/PyH5g"
        }
    }
}
```

Device authorization through mobile applications

In addition to initiating a regular OAuth authorization flow, mobile applications and single-page web applications can use the authentication API to initiate and complete the user authorization side of the OAuth device authorization flow.

There are a few differences between this case and the non-device case:

- You don't need to select Allow Authentication API Redirectless Mode on the Client window for the OAuth device client because the mobile or single-page web application doesn't receive tokens at the end of the flow.
- For the same reason, in the case of web applications, you don't need to enable Allow Redirectless Mode on the Authentication Application window when configuring the authentication API application.
- The initial request is made to the user authorization endpoint `/as/user_authz.oauth2` rather than `/as/authorization.oauth2`. As with the non-device flow, you must specify `pi.flow` for the `response_mode`. Optionally, the initial request can also provide the `user_code`. This endpoint doesn't need any other parameters.
- At the end of the flow, the `OAUTH_DEVICE_COMPLETED` state is returned to the API client. This response doesn't include an authorization code or tokens.

As with the non-device flow, you must select Bypass Authorization Approval on the Client window for the device client because the PingFederate authentication API does not yet support the OAuth consent approval step.

The models and actions for the `OAUTH_DEVICE_USER_CODE_REQUIRED`, `OAUTH_DEVICE_USER_CODE_CONFIRMATION_REQUIRED`, and `OAUTH_DEVICE_COMPLETED` states are documented in the Authentication API Explorer under the PingFederate Core adapter.