PingFederate Server

Contents

Release Notes

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. PingFederate extends employee, customer, and partner identities across domains without passwords, using only standard identity protocols: Security Assertion Markup Language (SAML), WS-Federation, WS-Trust, OAuth, and SCIM.

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

PingFederate 9.3.3 - December 2019

PingFederate 9.3.3 is a cumulative maintenance release for PingFederate 9.3, which introduced many new features, such as authentication API, Identifier First Adapter, notification publishers, CIBA support, Amazon CloudHSM support, a streamlined setup experience for PingID VPN use case, and more. For a full summary of the 9.3 release, see *PingFederate 9.3 - June 2019*.

Resolved issues

Ticket ID	**Description**
PF-24839	The `SameSite` attribute is now added only to secure cookies.
PF-24826	Resolved an issue causing session keys to be stranded on the CloudHSM, causing it to run out of key space and preventing PingFederate from performing any other verifications.
PF-24766	Resolved a potential connection leak issue when using outbound provisioning with PingDirectory.
PF-24671	Resolved an issue that caused single sign-on failures when the HTML Form Adapter's "remember my username" option was enabled and the username contained a space.
PF-24667	The OpenID Connect max_age parameter now accepts a value of 0.

PingFederate 9.3.2 - November 2019

PingFederate 9.3.2 is a cumulative maintenance release for PingFederate 9.3, which introduced many new features, such as authentication API, Identifier First Adapter, notification publishers, CIBA support, Amazon CloudHSM support, a streamlined setup experience for PingID VPN use case, and more. For a full summary of the 9.3 release, see *PingFederate 9.3 - June 2019*.

Resolved issues

Ticket ID	**Description**
PF-24638	Improved scheme validation of the `redirect_uri` parameter in browser-based OAuth flows.
PF-24604	Changes made through the `/oauth/cibaServerPolicy/settings` API endpoint are persisted through PingFederate restart.

Ticket ID	Description
PF-24564	Resolved an issue where the requests without the `User-Agent` header caused PingFederate to return an error when SameSite Safari filtering is enabled.
PF-24532	The `sameSiteSpecifier` is now set to `None` by default in `jetty-runtime.xml`.
PF-24511	Resolved an issue that caused users who had just changed their passwords to be prompted to log in with the error, `Your password must be reset before you can log on. Please change your password and try again or contact your system administrator.` This issue occurred only in clustered environment.
PF-24452	Resolved an issue where a policy tree could be executed with no acceptable mapping configured.
PF-24425	Resolved an issue causing PingDirectory user credential validation to fail when the service account did not have the `config-read` privilege.
PF-24340	Resolved an issue with state handling in the RADIUS Username Password Credential validator.
PF-24329	Changed the reference number generation algorithm to avoid the potential formation of words. Now, an 8-character, purely numeric number is used. The reference number is displayed to users for system errors.
PF-24292	Resolved an issue resulting in a handshake failure for some transactions during configuration replication.
PF-24124	Using the administrative console to update the metadata on an existing IdP connection now correctly updates the signature verification certificates.
PF-24112	Resolved an issue where the default Authentication Application was incorrectly invoked in the HTML Form Adapter's 'Trouble Signing On' flow, even when there was no Authentication Application configured in the Authentication Policies.
PF-24010	Resolved an issue where the Authentication API Explorer was not masking the password in some cases.
PF-23969	When configuring authentication policy sessions, using an integer up to the maximum of 43200 in the Session Revocation Lifetime (Minutes) field no longer causes an integer overflow.

PingFederate 9.3.1 - August 2019

PingFederate 9.3.1 is a cumulative maintenance release for PingFederate 9.3, which introduced many new features, such as authentication API, Identifier First Adapter, notification publishers, CIBA support, Amazon CloudHSM support, a streamlined setup experience for PingID VPN use case, and more. For a full summary of the 9.3 release, see *PingFederate 9.3 - June 2019*.

Resolved issues

Ticket ID	Description
PF-23950	To improve security, cluster encryption is enabled by default when cluster authentication is enabled.

Ticket ID	Description
PF-23895	The administrative API PUT/POST calls for WS-Federation connections no longer change `abortSsoTransactionAsFailSafe` from true to false.
PF-23863	The **Attribute Source ID** field on the **Attribute Sources & User Lookup** screen is now validated to ensure that it contains only alphanumeric characters.
PF-23851	Resolved an issue with connection configuration containing duplicate certificate aliases when upgrading from a release prior to PingFederate 8.4.
PF-23831	Multi-value entries used in custom data sources' filters are now correctly parsed.
PF-23797	PingFederate now properly encodes client IDs and client secrets when making OIDC requests to third-party OIDC Providers using the Authorization Code flow. This fix resolves interoperability issues with OIDC Providers such as Google and Keycloak that use the hyphen character ("-") in client IDs.
PF-23713	Using a selector that doesn't include any result values no longer causes the **Authentication Policies** page in the administrative console to crash when viewing the policy.
PF-23617	Users are now re-challenged after entering an incorrect challenge response when using the RADIUS Username PCV.
PF-23541	Improved memory utilization in the Client-Initiated Backchannel Authentication workflow.
PF-23540	PingFederate no longer uses the LDAP `ServerSideSortRequestControl` without first verifying whether it is supported.
PF-23528	The `Your password is expired and must be changed` error message no longer continues to display on the login screen after the user has successfully changed their password.
PF-23512	Password changes no longer fail when the underlying LDAP PCV is using Oracle Directory Server Enterprise Edition (ODSEE) and the user's password has expired.
PF-23506	PingDirectory clustered environments that have latency no longer cause a possible null value to be received when searching for an OAuth client immediately after it has been created.
PF-23464	The OpenToken Adapter (now version 2.5.8) now provides an option to URL encode the value of the extended cookie to ensure compliance with RFC 6265.
PF-23434	PingFederate now maps the `login_hint` parameter from incoming OIDC requests to the **Incoming User ID** option's `Requested User` attribute from the Context source in Authentication Policies.
PF-23383	The selected stored procedure is no longer cleared when a user clicks **Next** on the **Stored Procedure Location** tab. This problem occurred when JIT provisioning had previously been configured from an SQL datastore using a stored procedure, and the datastore had multiple schemas.
PF-23303	PingFederate no longer ignores the global default policy for password expiration notifications when using PingDirectory or Oracle Directory Server Enterprise Edition (ODSEE). Previously, the notifications were displayed only when a user had a password policy explicitly set using the `ds-pwp-password-policy-dn` attribute.
PF-23018	The SP Adapter to IdP Connection mapping dependency error is now removed when the extended contract on the adapter is deleted.

Ticket ID	Description
PF-23014	PingFederate as an IdP no longer sends SLO requests to partners that only support SP-init SLO (not IdP-init SLO).

PingFederate 9.3 - June 2019

Enhancements

Authentication

Authentication API

The end user experience is critical to identity services for any organization, but security cannot be sacrificed. Typically, the end user experience is designed and maintained by front end developers while the security policy is administered by identity architects. Requiring the two groups to coordinate changes adds friction to the speed at which new changes can be applied. To eliminate this friction, PingFederate now includes an authentication API. The authentication API separates the presentation layer from the security policy. Authentication policy continues to be maintained in PingFederate while the UI end users interact with can be maintained in a separate web application. The authentication API directs the presentation layer through the credential collection prompts of the authentication workflow. This allows the identity architect to change authentication policy without requiring code changes to the presentation layer, and the front end developer can release changes to the end user UI without waiting on a change to PingFederate.

PingOne® for Enterprise setup

The setup wizard for connecting PingFederate to PingOne for Enterprise has been updated to improve usability and to include support for the PingID® VPN use case. SSO to PingOne for Enterprise and PingID for VPN can be individually enabled. PingFederate can also be configured to automatically generate sample users when evaluating SSO via PingOne for Enterprise.

Identifier First Adapter

When a variety of user types are authenticating at PingFederate, it may not be appropriate to challenge all users with a default credential type, such as a password. It may be better to ask the user for their identifier first and then determine the user population and the enrolled credential types. The new Identifier First Adapter provides a way for PingFederate to collect the user identifier and then determine how to challenge the user for credentials. The determination could be based on an email address suffix, which indicates the identity provider, or a data source query that returns the authentication methods that apply to the user.

Extended Properties

It’s common for data to exist about applications and partners beyond the configuration PingFederate requires for transaction processing. Rather than storing this information in external systems, it’s now possible to define a schema of Extended Properties in PingFederate to store this information with OAuth client and connection definitions. This enhancement builds on top of, and replaces, the Extended Client Metadata previously available.

Extended Properties Authentication Selector

Authentication policy is at the core of secure access to applications. Once defined and tested, an authentication policy should change very infrequently. When authentication requirements for an application require an update, it is better to represent such update as a change to the nature of the application than as a change to a shared set of authentication policies. As a complement to extended properties, a new authentication selector has been added to determine the authentication required based on the extended properties defined in SP connections and OAuth clients.

REST API data store

A new data store type has been added to enable the querying of attribute data from JSON-based REST APIs.

Authentication policies for forgot password

Forgotten password recovery, or password reset, is a prime target for account takeover attacks. It's important that the authentication required to recover from a forgotten password is sufficiently strong, relative to the strength of the password credential itself and any authentication policies that include password authentication. In other words, password reset cannot be a weak link in the overall authentication policy. The authentication options available for password reset have been extended to support any authentication policy configured in PingFederate.

Directory server password policy messaging

Users need feedback when they attempt to set a new password, but that password does not meet the requirements of the underlying directory. The HTML Form Adapter has been updated to propagate the password policy violation details from the directory server directly to the presentation layer. This improves the user experience as the messaging is dynamic and can change over time as changes are made to the directory server. This messaging uses the existing localization framework for end user UIs so that the error messages can be translated into the user's preferred language.

HTTP Request Parameter Authentication Selector enhancements

The HTTP Request Parameter Authentication Selector now offers additional result-matching options. In addition to selector results that branch authentication policy for specific parameter values, it's now possible to branch authentication policy when the parameter matches any configured selector result, no configured selector result, or is not included in the request. Previously, authentication selector processing would fail, and the next applicable authentication policy would be executed under these conditions.

Requested AuthnContext Authentication Selector enhancements

The Requested AuthnContext Authentication Selector also offers new result-matching capabilities. In addition to selector results that branch authentication policy for specific requested authentication contexts, it's now possible to branch authentication policy when a request does not include an authentication context or includes one that does not match any configured authentication context. Previously, authentication selector processing would fail, and the next applicable authentication policy would be executed under these conditions.

Authentication policy sequence logging

Authentication policies can enable very sophisticated workflows, involving many steps and contextual rule evaluations. Authentication policy logging has been enhanced to make it easier to track every step in the workflow and to simplify workflow debugging. This new set of log messages are associated with a new category for easy filtering and written in a structured format for automated processing.

Authentication source session logging

Understanding the life cycle of authentication sessions can help analyze user experience and their exposure to application access. Knowing the time allotted to users may help during security reviews to ensure access isn't given for a time period beyond what is necessary. PingFederate now captures authentication life cycle events in the security audit log.

OAuth

Client Initiated Backchannel Authentication (CIBA)

Client Initiated Backchannel Authentication (*openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html*) is an extension to OpenID Connect that is gaining interest by organizations that want to improve the end-user experience during authentication and authorization in a federated environment. This extension defines a new OAuth grant type where user consent can be requested through an out-of-band flow. For example, CIBA improves the user experience when making an online purchase from a merchant as it doesn't 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.

Option to require PKCE

Proof Key for Code Exchange (PKCE) is a technique to mitigate the risk of authorization code interception attacks (see *https://tools.ietf.org/html/rfc7636*). While PingFederate supports PKCE, participation is voluntary as initiated by OAuth clients. Administrators can now make PKCE mandatory and reject authorization requests from clients who fail to comply.

Scope group expansion

Scope groups provide a convenient mechanism for organizing sets of scopes that can be requested by OAuth clients by individual group names. Scope groups can be beneficial for a number of reasons. One example is to provide an abstraction layer between the scope group name that a client can use to request authorization and the corresponding scopes that can be authorized. This gives administrators the flexibility to modify a set of scopes available to a client without having to update that client to be aware of such changes. However, resource servers may not be aware of the individual scopes that compose the scope group. A new option has been added to expand scope groups to their composing scope list in access tokens.

Access Token validation optimization

Internally Managed Reference Token validation has been optimized to reduce response time when a large number of Access Token Managers are defined.

Grant expiration tracking in ID token

It is now possible to include a claim in the ID token to track the expiration time of the corresponding authorization grant.

Pairwise Pseudonymous Identifiers

Some federations require user anonymity so that different relying parties are unable to collaborate over individual users. It's now possible to enable Pairwise Pseudonymous Identifiers, as defined by the OpenID Connect specification, for OpenID Connect clients.

System

Notification Publishers

Some events that PingFederate processes require out-of-band communication with users; for example, notifying administrators about certificate expirations and notifying end users about password management events. Some types of users or events may require different delivery methods. The notification system in PingFederate has been overhauled to provide a pluggable architecture to publish notifications in a variety of ways, configurable based on the types of events and users. In this release, PingFederate includes both SMTP and AWS Simple Notification Service (SNS) publishers out of the box. Existing SMTP configuration data will be migrated to this new model upon upgrade.

Automatic configuration replication upon monitored metadata changes

PingFederate can monitor metadata published by SAML partners. This is used to detect changes to partner metadata, typically a result of rotating keys used to sign and verify SAML signatures. In previous versions, PingFederate would download changes to metadata to the administrative server and notify the administrator of the change. Certain changes, such as changes to the list of certificates, are now automatically replicated to the runtime servers in the cluster.

Remove unused certificates from connection upon monitored metadata changes

Rotating certificates can lead to a growing list of inactive ones. PingFederate now automatically deletes verification certificates from connection configuration when they no longer are included in a partner's SAML metadata.

Master Key secured by AWS Key Management Service

PingFederate cryptographically secures certain data elements using a Master Key (found in the `pf.jwk` file), which is normally unique per installation. During the initial startup, PingFederate automatically generates a randomized key. Customers that deploy new configurations using a pipeline that provisions an entirely new PingFederate cluster (i.e., blue/green) can include a `pf.jwk` file, so the new infrastructure can read data that was previously secured by a different cluster. Customers running in AWS now have the option to secure the `pf.jwk` file using Amazon KMS.

Connection and OAuth Client name auditing

In addition to connection IDs and OAuth client IDs, connection names and client names are now available for security audit logging.

Support for Amazon Corretto Java 8 and 11

This release includes support for Amazon Corretto Java 8 and 11. Amazon Corretto is a no-cost distribution of OpenJDK provided by Amazon with a long-term support commitment for performance and security enhancements. Note that while other combinations may function as expected, the only supported cluster configurations are those where each node utilizes the same version and distribution of Java.

Administrative API enhancements

To simplify batch processing of administrative API calls, the following APIs have been enhanced to allow for the object ID to be declared by the API client rather than automatically generated:

- `/authenticationPolicyContracts`
- `/keyPairs/signing/import`
- `/keyPairs/signing/generate`
- `/keyPairs/sslClient/import`

- `/keyPairs/sslClient/generate`
- `/keyPairs/sslServer/import`
- `/keyPairs/sslServer/generate`
- `/dataStores`

Additionally, the administrative API has been extended to manage certificate revocation settings at `/certificates/revocation`.

Other improvements

- AWS CloudHSM has been added as a supported HSM.
- Renamed OAuth Device Flow to OAuth Device Authorization Grant to align with the change in the specification.
- The HTML Form Adapter's alternate authentication source button for FIDO via PingID has new styling to reflect the FIDO method available to the user given the browser and OS they are using.
- Gemalto SafeNet Luna Network HSM client 7.2 and the Gemalto Safenet Luna eLab environment has been qualified.
- A warning message is now logged when upgrading PingFederate from a previous version when the tcp.xml file is not merged.
- PingFederate has been qualified on Alpine Linux.
- PingFederate has been qualified on Amazon Linux 2.
- Updated the following bundled components and third-party dependencies:
  - PingID Adapter 2.5.1
  - PingID Connector 1.0.1
  - PingID PCV (with integrated RADIUS server) 2.5.0
  - Jetty 9.4.18
  - Log4j2 2.11.2
  - UnboundID LDAP SDK 4.0.11

Resolved issues

Ticket ID	Description
PF-23012	Resolved an issue where an error occurred when a null value was returned for a pseudonym attribute from an adapter.
PF-22962	The administrative console no longer displays an error message even when a unique column is selected as part of the just-in-time provisioning configuration.
PF-22947 and PF-21215	Resolved a couple of performance issues and improved the throughput of PingFederate as a result.
PF-22846	Fixed an issue where AWS_PING failed when the access_key parameter or the secret_key parameter was removed from the `tcp.xml` file.
PF-22419	Resolved an issue where PingFederate did not preserve query parameters sent to the Account Recovery endpoint (`/ext/pwdreset/Identify`) when those query parameters were configured as tracked parameters.
PF-22191	Fixed an issue where a unique ID field could not be set in an identity profile when it was configured with profile management without registration.

Ticket ID	Description
PF-21042	An administrator, who signed on to the console by using a directory account, can no longer initiate a change password request for a local account with the same username. (It is worth mentioning that this issue only impacted the administrative console experience; it was not possible to change the password of the local account.)

Known issues

Administrative console and administrative API

For 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.

Known limitations

SameSite cookie handling in Chrome

Note: To implement this change, you must be running PingFederate 9.3.1 or above.

In Chrome release 80, it is expected that the default behavior of cookies that do not have a `SameSite` specifier will change. Cookies without the `SameSite` value specified, are expected to have `SameSite=Lax` set by default.

Enabling the `Lax` value makes cookies available to third-parties through HTTP GET requests, but not by other methods, such as POST. This capability can also be enabled in other browsers such as Firefox by enabling a user setting.

When the `Lax` value is enabled, cookies without the `SameSite` specifier are restricted to the same site the user is browsing. The impact to PingFederate users, when this setting is enabled, is that they will need to re-authenticate existing SSO sessions.

If cookies need to be available in a third-party context, you must configure the `SameSite=None` setting. You can do this by uncommenting the `<Set name="sameSiteSpecifier">None</Set>` line in the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file.

Administrative console and administrative API

- 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 via 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 plug-in, 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 screens in the PingFederate console.
- If authenticated to the PingFederate administrative console using certificate authentication, a session that has timed out may 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 once authentication is complete. Similar behavior applies for certificate authentication, in principle. However, because the browser may automatically resubmit the certificate for authentication, the browser may redirect to the administrative console and not the login page.

Hardware security modules (HSM)

- When using PingFederate with an HSM from Gemalto or nCipher, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.
- PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.

SSO and SLO

- 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

- 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

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

OAuth

PingFederate does not support 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`.

It is worth noting that while it is 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 (if implemented), we recommend *not* to do so to avoid record migration issues if it is decided later that client records should be stored in a directory server.

Customer identity and access management

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

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.

Logging

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, then the USER_KEY attribute will not be masked in the server logs. Other persistent grant attributes will be masked.

Database logging

If PingFederate cannot establish a JDBC connection at startup, PingFederate will continue to write log messages to the failover log file, despite the failover and resume configuration. When the JDBC connectivity issue is resolved, restart PingFederate. On restart, PingFederate will start writing log messages to the database.

Note that if PingFederate is able to establish a JDBC connection at startup, PingFederate will be able to write log messages to the failover log when it encounters a JDBC connectivity issue *and* resume writing log messages to the database when it re-establishes the JDBC connection.

RADIUS NAS-IP-Address

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.

Configcopy

- When the **`configcopy`** tool is used to copy all connections, channels, data sources, adapters, or token translators, overridden properties are applied to all instances. Care must be taken when applying overrides for copy-all operations.
- The **`configcopy`** tool supports copying only a single reference for each of the following configuration items that are defined for a given connection: adapter, data source, Assertion Consumer Service URL, Single Logout Service URL, and Artifact Resolution Service URL. When multiple adapters, data stores, or any of the aforementioned service URLs are associated with a given connection, only the first reference to each is copied.
- The **`configcopy`** tool does not support creation of configuration data that does not exist in the source. If an override parameter for a parameter that does not exist in the source configuration is set, the behavior of the target system is not guaranteed.
- The **`configcopy`** tool, when used for copying plug-in configurations (including adapters, token translators, and other data stores), does not currently support overrides of complex data structures, including tables, extended contract attributes, and masked fields.
- When the **`configcopy`** tool is used to copy connection data, any SOAP SLO endpoints defined in the source are not copied to the target, even if the SOAP SLO endpoint is the only SLO endpoint defined at the source. These must be manually added to the target.

Deprecated features

Email configuration

Starting with version 9.3, PingFederate provides a pluggable architecture to publish notifications in a variety of ways, configurable based on the types of events and users. PingFederate 9.3 includes both SMTP and AWS Simple Notification Service (SNS) publishers out of the box. As a result, the upgrade process now migrates email server settings from the source installation as an SMTP Notification Publisher instance in the new PingFederate installation. To manage notification publisher instances, go to the **System# Notification Publishers** screen.

Oracle Solaris 10

Starting with version 9.2, Oracle Solaris 10 is no longer included in the PingFederate qualification process. For a list of supported operating systems, see *System requirements* on page 60.

Auto-Connect™

The Auto-Connect™ feature has been discontinued in PingFederate 9.2. If such usage is detected in the source installation during an upgrade, the upgrade tool warns the administrator about it. Administrators should remove Auto-Connect™ configuration from the source installation and re-run the upgrade tool.

JMX monitoring support for outbound provisioning

JMX monitoring of outbound provisioning is no longer an option starting with version 9.2. If such usage is detected in the source installation during an upgrade, the upgrade tool warns the administrator about it. No further action is required. Outbound provisioning transactions are written to the `provisioner-audit.log` file in the new installation (see *Outbound provisioning audit logging* on page 234).

Logging configuration

The default logging configuration has been vastly optimized since PingFederate 8.2. As a result, the product distribution no longer includes the `terse.example.log4j2.xml` file in the `<pf_install>/pingfederate/server/default/conf` directory, starting with version 8.4.

For more information, see *Log4j 2 logging service and configuration* on page 225.

Plain text email notification templates

Starting with version 8.2, PingFederate has switched the format of its email notification from plain text to HTML. The new HTML-based templates (`message-template-*.html`) are located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory. As a result, PingFederate no longer maintains the plain text templates (`message-template-*.txt`).

To preserve previous modifications (if any), you must migrate custom changes manually. For more information, see *Copy customized files or settings* on page 963.

SpSessionAuthnAdapterId and SourceResource (query parameters for the /sp/startSLO.ping endpoint)

Support for the previously optional SpSessionAuthnAdapterId and SourceResource have been dropped in favor of the SLO improvements introduced in version 8.2.

BoneCP as the JDBC 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 more recent) 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.

DSA certificate creation

Starting with PingFederate 7.3, it is no longer possible to create DSA key pairs in the certificate management pages of PingFederate. Import of DSA key pairs continues to be supported.

Username Token Translator

As of PingFederate 7.2, the Username Token Translator has been deprecated and replaced with an integrated Username Token Processor. While the integrated Username Token Processor and the deprecated Username Token Translator may be simultaneously deployed, it is recommended to migrate to the new token processor.

For instructions, see *Migrate to the integrated Username Token Processor* on page 971.

LDAP Adapter

Starting with PingFederate 7.2, the LDAP Adapter is no longer supported. This adapter was deprecated in PingFederate 6.6 and replaced by the LDAP Username Password Credential Validator (PCV), which can be used with the HTML Form or HTTP Basic Adapters.

Upgrade considerations

Several specific modifications made since PingFederate 8.0 may affect existing deployments. Please review.

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. For setup information, see *Integrate with Gemalto SafeNet Luna Network HSM* on page 94.

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* on page 277 for more information.

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.

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 *Configure validation for the AudienceRestriction element* on page 289.

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. For more information, see *Update custom authentication selector* on page 970.

Provisioning data store 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 has since been 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 9.3, 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 9.3, you must use the **`provmgr`** command-line tool to reset the provisioning data store on the upgraded installation. For more information, see *Review database changes* on page 965.

Security enhancement in JDBC data store queries

A security enhancement has been made in PingFederate 9.0 to safeguard JDBC data store queries against back-end SQL injection attacks. This protection is enabled for all new installations. For upgrades, see *Review database changes* on page 965.

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 *Copy customized files or settings* on page 963 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 on modifying this index in your existing table, see *Review database changes* on page 965.

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`) via 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 could be re-enabled. Refer to *this knowledge base article* 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, to 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 *Configure cluster protocol properties* on page 913.

Metadata signing

Previously, when no signing certificate was chosen in the **System# Metadata Settings# Metadata Signing** screen, 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 in the **Security# Signing & Decryption Keys & Certificates** screen.

Starting with PingFederate 8.1, if no certificate is selected in the **Metadata Signing** screen, PingFederate provides unsigned metadata at both aforementioned endpoints. Select a certificate in the **Metadata Signing** screen 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 for greater 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 more recent) 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 *Review database changes* on page 965.

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 *Review log configuration* on page 969 for instructions.

Note: PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see . To obtain your database driver JAR file, contact your database vendor. Database driver file should be installed to the `<pf_install>/pingfederate/server/default/lib` directory. You must restart the server after installing the driver.

Upgrade considerations introduced in PingFederate 7.x

Hostname verification for LDAPS

For LDAP type data stores with LDAPS enabled, hostname verification of the certificate is enabled by default for all new data stores staring with PingFederate 7.3. When upgrading from a previous version of PingFederate, this option is disabled for existing data stores 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 data store. Follow the instructions in *Review database changes* on page 965 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 *Review database changes* on page 965.

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 *Review database changes* on page 965).

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. For more information, see *HTML Form Adapter* on page 716.

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 *Review database changes* on page 965.

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 may 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 **OAuth Server# Client Management** screen 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 may 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 may 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 may 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 such connections to use the newer algorithm RSA-OAEP (see *Select an encryption certificate* on page 541 for SP connections and *Choose an encryption certificate (SAML 2.0)* on page 660 for IdP connections).

OAuth persistent grants expiration

When upgrading to PingFederate 6.8 (or a more recent version), 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 **OAuth Server# Client Management** screen 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.

Previous releases

For information about enhancements and issues resolved in previous major and minor releases of PingFederate, follow the links below to their release notes. From there you can also access the release notes for older maintenance releases.

- *9.2 December 2018*
- *9.1 June 2018*
- *9.0 December 2017 and previous releases*

Get Started with PingFederate Server 9.3

This guide provides information about getting started with Ping Identity® 's PingFederate to deploy a secure Internet-identity platform, including single sign-on (SSO) based on the latest security and e-business standards.

Introduction to PingFederate

Welcome to PingFederate, Ping Identity® 's enterprise identity bridge. PingFederate enables outbound and inbound solutions for single sign-on (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 only standard identity protocols (Security Assertion Markup Language—SAML, WS-Federation, WS-Trust, OAuth and OpenID Connect, and SCIM).

About identity federation and SSO

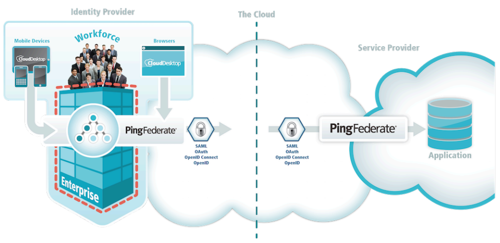
Federated identity management (or *identity federation*) enables enterprises to exchange identity information securely across domains, providing browser-based SSO. Federation is also used to integrate access to applications across distinct business units within a single organization. As organizations 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* on page 30.

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, for example, might be an enterprise that manages accounts for a large number of users who may 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.



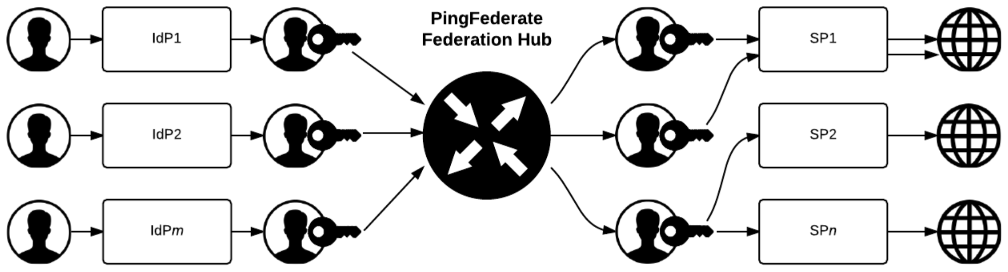
MY TITLE Secure single sign-on

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.

PingFederate provides complete support for both roles. Note that business processes of a single organization might encompass both SP and IdP use cases; this scenario can be handled by a single instance of PingFederate.

Federation hub

To most organizations, identity federation means negotiating and managing federation settings with partners. As the number of partners grows, so does the administrative overhead. In addition, different federation protocols may also hinder application development and SSO implementation. To remove these obstacles, PingFederate can be configured as a *federation hub* to extend federated access across partners supporting different federation standards, SAML and WS-Federation for example, as well as to provide a centralized console to simplify SSO administration. By bridging the identity providers and service providers through the federation hub, administrators also have the option to multiplex a single connection for multiple partners, adding additional use cases and reducing administration and implementation costs.

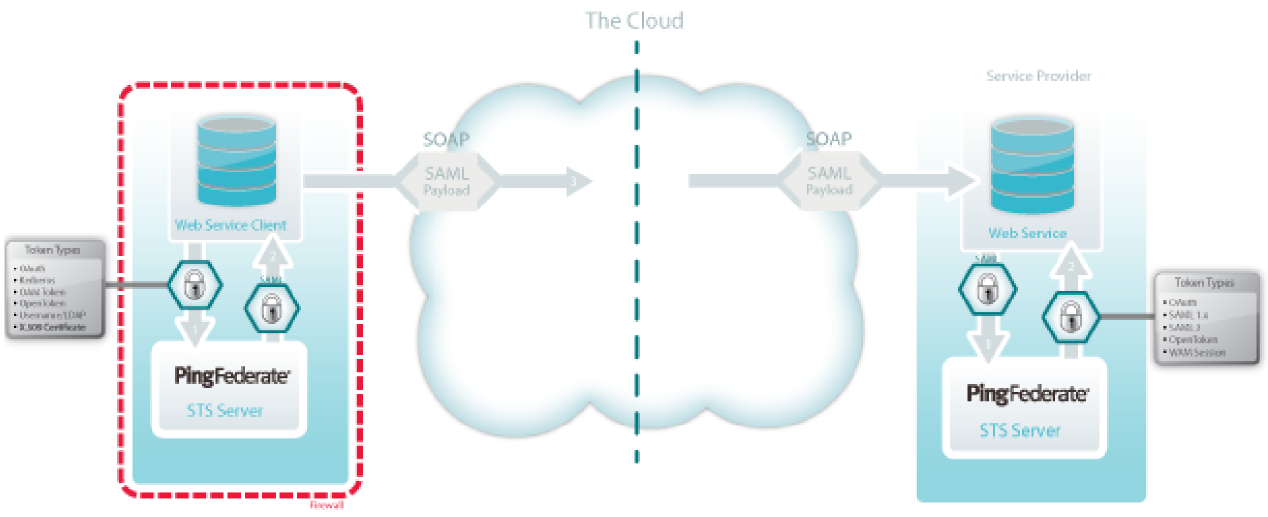


Security token service

The PingFederate WS-Trust Security Token Service (STS) allows organizations to extend SSO identity management to web services. (For information about WS-Trust and the role of an STS, see *Web services standards* on page 47).

The STS shares the core functionality of PingFederate, including console administration, identity and attribute mapping, and certificate security management. With PingFederate, web services can securely identify the end user who has initiated a transaction across domains, providing enhanced service while simultaneously ensuring appropriate information access and regulatory accountability.

PingFederate can be used 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 *About WS-Trust STS* on page 100).



MY TITLE 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. 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 Representational State Transfer (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 SaaS provider.

(For information about OAuth and the role of an AS, see *OAuth 2.0 and PingFederate AS*.)

The PingFederate OAuth AS can be configured independently or in conjunction with STS and browser-based SSO for either an IdP or an SP deployment. For more information, see *About OAuth* on page 103.

Note: OAuth AS capabilities may require additional licenses. For more information, please contact sales@pingidentity.com.

User account management

In an identity federation, accounts are maintained for users at the IdP site. However, an SP will often have its own set of user accounts, some of which may correspond to IdP users. The SP may also need to establish and maintain parallel accounts for remote SSO users to enforce authorization policy, customize user experience, comply with regulations, or a combination of such purposes.

To facilitate cross-domain account management, PingFederate provides two kinds of user provisioning for browser-based SSO, one designed for an IdP and one for an SP:

- At an IdP site, an administrator can automatically provision and maintain user accounts for partner SPs who have implemented the System for Cross-domain Identity Management (SCIM) or, when optional plug-in SaaS Connectors are used, for selected hosted-software providers.
- At an SP site, an administrator can provision accounts within the organization automatically from SCIM-enable IdPs or use information from SAML assertions received during SSO events.

For more information, see *User provisioning* on page 125.

Enterprise deployment architecture

With PingFederate's enterprise-deployment architecture, all protocol definitions, public key infrastructure (PKI) keys, policies, profiles, etc., are managed in a single location, eliminating the need to maintain redundant copies of these configurations and trust relationships. Furthermore, when new protocols, profiles, or use cases need to be added, you only have to configure them once to make them available to your entire organization.

PingFederate also improves security by creating a single “doorway” in your perimeter through which all identity information must travel. Using PingFederate, all of your internal users who sign on to external applications exit through this doorway, while all external users who sign on to your internal systems enter through the same doorway.

The single-doorway approach also provides 100 percent visibility to all federation activities. The extensive auditing and logging capabilities of PingFederate enable you to satisfy all of your logging-related compliance and service-level requirements from a single location, as opposed to having to acquire and consolidate disparate logs from throughout your organization.

Use case configuration

By providing a single configuration paradigm supporting different protocols, PingFederate reduces complexity and learning curves. Furthermore, the step-by-step administrative console minimizes the potential for errors by guiding administrators through configuration steps applicable only to the business use cases they need to support.

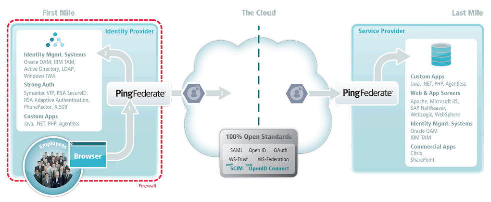
ⓘ **Tip:** For IdPs, connection templates that automatically configure many steps in the administrative console are available for several use cases, including setting up SSO connections to selected SaaS vendors. (For more information, see *Outbound provisioning for IdPs* on page 125).

Additional features

PingFederate lightweight, standalone architecture means you can receive the benefits of standards-based SSO and API security integration without the cost and complexity of deploying a complete identity management (IdM) system. The PingFederate server integrates and coexists with existing home-grown and commercial IdM systems and applications, using these key features available separately from Ping Identity.

Integration kits

PingFederate provides a suite of integration kits to complete the first- and last-mile integration with your existing IdM systems and web applications. PingFederate integration kits are available for download from the Ping Identity Downloads website, take only minutes to install, and are configured from within the PingFederate administrative console.



MY TITLE Multiple security-domain, multi-protocol federation

Integration kits enable rapid session integration with both existing authentication services and target applications. In addition, PingFederate includes a Software Development Kit for creating custom integrations.

For more information, see *SSO integration kits and adapters* on page 112.

Token translators

Ping Identity offers special token processors (for an IdP) and token generators (for an SP) to enable the WS-Trust STS to validate and issue a variety of token types. These plug-ins, which supplement built-in SAML token processing and generation, are designed to handle local identity tokens required in a variety of security contexts.

For more information, see *Token processors and generators* on page 100.

SaaS connectors

SaaS connectors offer a streamlined approach for browser-based SSO to selected SaaS providers, including automatic user provisioning and deprovisioning (see *Outbound provisioning for IdPs* on page 125). The Connector packages (available separately) include quick-connection templates, which automatically configure endpoints and other connection information for each provider.

Cloud identity connectors

Ping Identity offers social identity integration with social networking sites. The OpenID cloud-identity connector leverages OpenID 2.0 social networking providers (including Google and Yahoo!) for registration and access to cloud-based applications. Connectors for Twitter, LinkedIn, and Facebook leverage user logins for registration and access to cloud-based applications.

About PingOne

PingOne® for Enterprise is a cloud-based identity as a service (IDaaS) framework for secure identity access management. 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.

For more information on PingOne, please visit pingone.com.

Supported standards

PingFederate provides flexible, integrated support for the Security Assertion Markup Language (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

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, however, 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.

With OAuth 2.0 and OpenID Connect 1.0 support, PingFederate can be configured as an authorization server (AS), an OpenID Provider (OP), and a Relying Party (RP). (Note that OP and RP are the synonyms for IdP and SP, respectively.)

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 that authentication.

> ⓘ **Note:** The SAML subject may be a person, a web application, or a web server. Since the subject is often a person, the term *user* is generally employed throughout our documentation.

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 whether (or 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 it can be employed as a standalone server.

Authorization Server

An OAuth AS issues access tokens and refresh tokens to OAuth clients after the resource owner has fulfilled the authentication requirement.

OpenID Provider

An 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

The SAML specifications provide a system of building blocks and support components for achieving secure data exchange in an identity federation. These include:

- Assertions
- Bindings
- Profiles
- Metadata
- Authentication Context

Assertions

Assertions are XML documents sent from an IdP to an SP. Each assertion contains identifying information about a user who has initiated an SSO request.

Bindings

A SAML binding describes the way messages are exchanged using transport protocols. PingFederate supports the following bindings:

HTTP POST

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

HTTP Artifact

Describes how to use an artifact to represent a SAML message. The artifact can be transported via an HTML form control or a query string in the URL.

HTTP Redirect (SAML 2.0)

Describes how SAML messages are transported using HTTP 302 status-code response messages.

SOAP (SAML 2.0)

Describes how SAML messages are to be transferred across the back channel (Simple Object Access Protocol).

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 therefore 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. This information includes, for example, profile and binding support, connection endpoints, and certificate information.

Whether you are publishing or consuming 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 may want information surrounding how the user was originally authenticated by the IdP, in addition to the assertion itself. The SP may use 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 can create authentication-context declarations. Partners may 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: *saml-authn-context-2.0-os.pdf*). However, it is up to partners to decide if additional authentication context is required and if these classes supply an adequate description. For SAML 1.x, the authentication context (called *AuthenticationMethod*), if used, must be specified as a URI (see *oasis-sstc-saml-core-1.1.pdf*).

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

Alternatively, several PingFederate integration kits provide methods that can be used by the developer to insert authentication context from external IdP applications into the assertion. Conversely, the SP developer can call methods for extracting authentication context from an assertion. Ultimately, it is up to the SP developer and application to create access control or other processing based on the context.

Browser-based SSO

Browser-based SSO includes SAML 1.x, 2.0, WS-Federation, and OpenID Connect and provides standards-based SSO, Single Logout (SLO), Attribute Query and XASP, and the WS-Federation Passive Requestor Profile for SP-initiated SSO.

SAML 1.x profiles

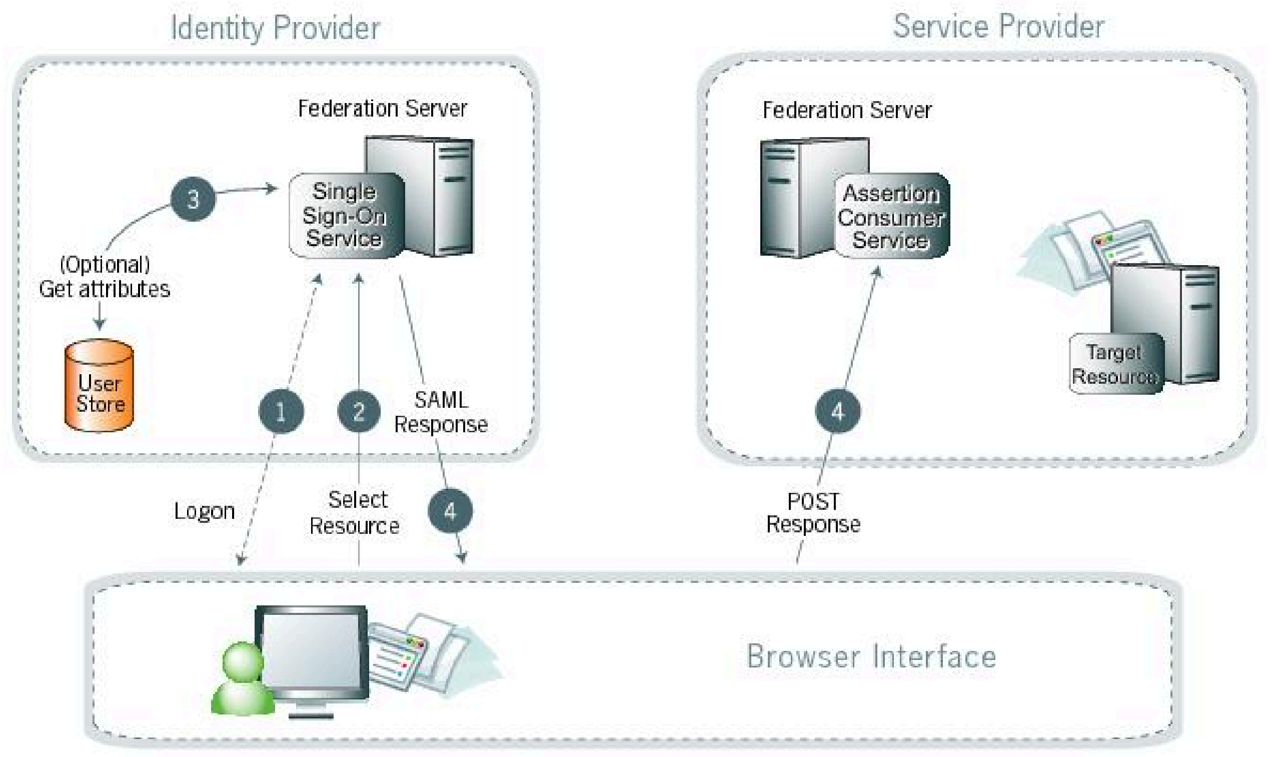
SAML 1.0 and 1.1 profiles provide for browser-based SSO, initiated by an IdP, using either the POST or artifact bindings.

In addition, the specifications provide for a non-normative SP-initiated scenario (called “destination-first”), which allows web developers to create applications that enable a user to initiate SSO from the SP site.

SSO—Browser-POST

About this task

In this scenario, 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 via HTTP POST.



MY TITLE SSO browser/POST profile

Processing steps:

Steps

1. A user has logged on to the IdP.

   (If a user has not yet logged on for some reason, he or she is challenged 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.

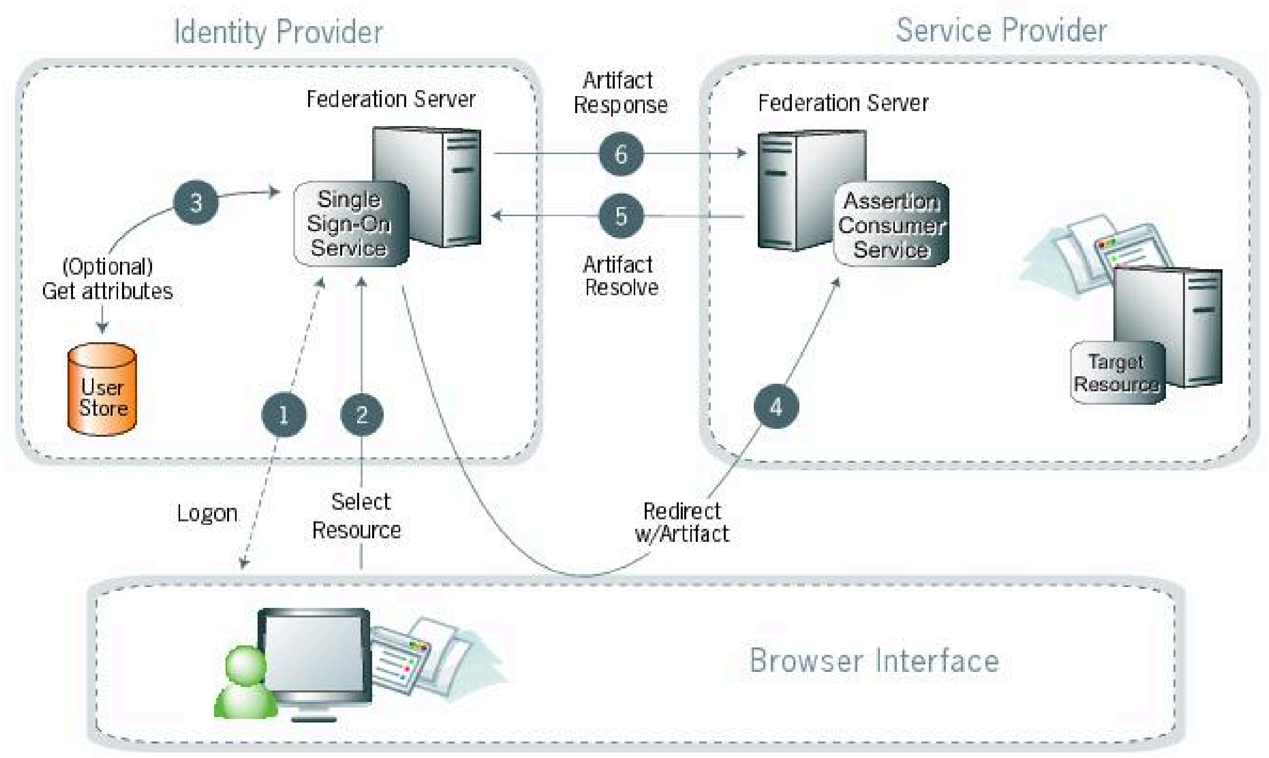
> **Note:** SAML specifications require that POST responses be digitally signed.

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

SSO—Browser-Artifact

About this task

In this scenario, the IdP sends a SAML artifact to the SP via either HTTP POST or a redirect (shown in diagram). The SP uses the artifact to obtain the associated SAML response from the IdP.



MY TITLE SSO browser/artifact profile

Processing steps:

Steps

1. A user has logged on to the IdP.

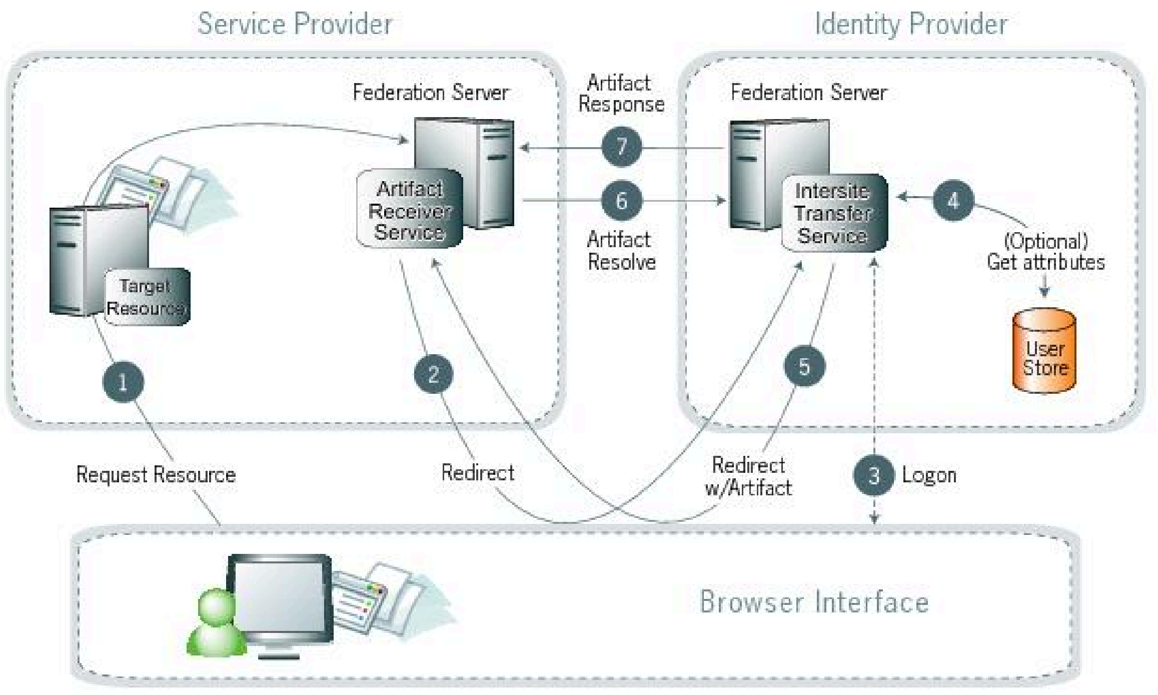
   (If a user has not yet logged on for some reason, he or she is challenged to do so at step 2).
2. The user clicks a link or otherwise requests access to a protected SP resource.
3. Optionally, the IdP retrieves attributes from the user data store.
4. The IdP federation server generates an assertion, 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 a valid assertion is received, the SP establishes a session and redirects the browser to the target resource.

SP-initiated (destination-first) SSO

About this task

In an SP-initiated (also known as destination-first) transaction the user is connected 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 federation agreement, authentication is managed by the IdP. The SP sends an authentication request to the IdP.



MY TITLE SP-initiated SSO

Processing steps:

Steps

1. The user requests access to a protected SP resource. The request is redirected to the federation server (for example, PingFederate) to handle authentication.
2. The federation server sends a SAML request for authentication to the IdP's SSO service (also called the Intersite Transfer Service).
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
4. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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 assertion is valid, the SP establishes a session for the user and redirects the browser to the target resource.

SAML 2.0 profiles

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

- Single Sign-On (SSO)
- Single Logout (SLO)
- Attribute Query and XASP
- IdP Discovery

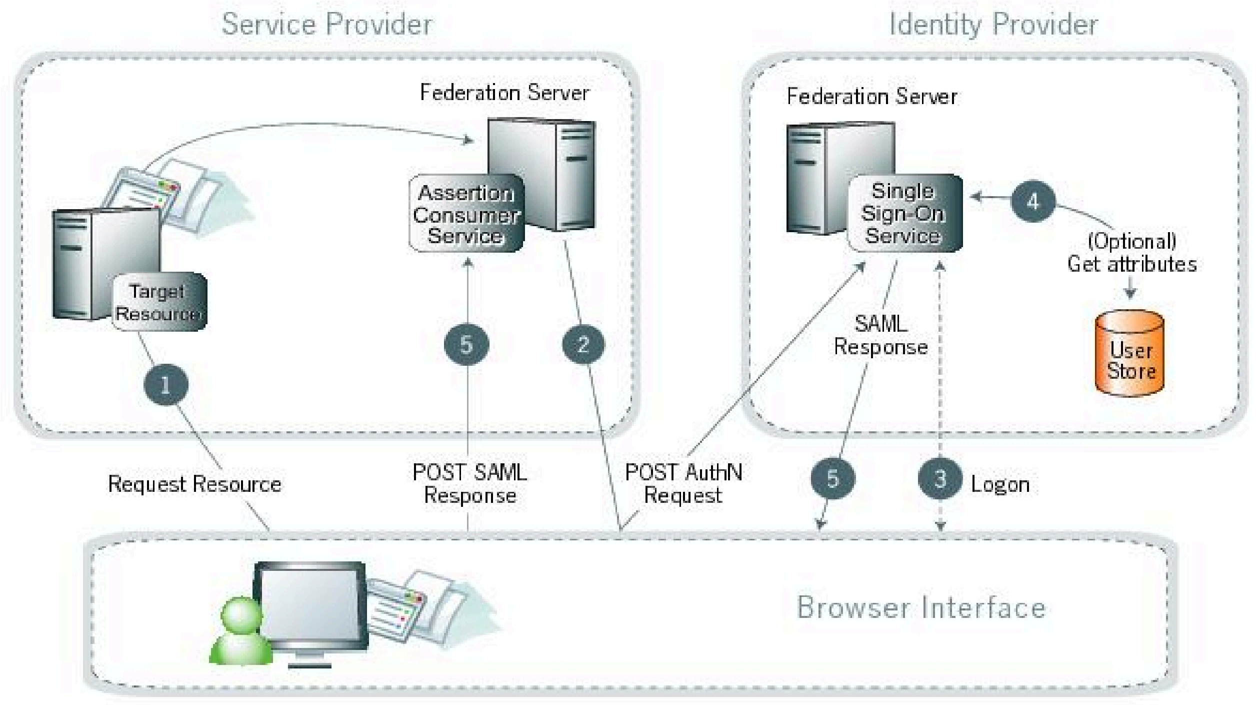
Single sign-on

SAML 2.0 substantially increases the number of possible SSO profile variations by fully enabling SP-initiated transactions. When SP- and IdP-initiated protocols are paired with transport binding specifications, the combinations result in eight practical SSO scenarios. (For more information, see subsequent topics.)

SP-initiated SSO—POST-POST

About this task

In this scenario a user attempts to access a protected resource directly on an SP website without being logged on. The user does not have an account on the SP site, but does have a federated account managed by a third-party IdP. The SP sends an authentication request to the IdP. Both the request and the returned SAML assertion are sent through the user's browser via HTTP POST.



MY TITLE SP-initiated SSO: POST/POST

Processing steps:

Steps

1. The user requests access to a protected SP resource. The request is redirected 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 is automatically posted to the IdP's SSO service.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
4. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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.

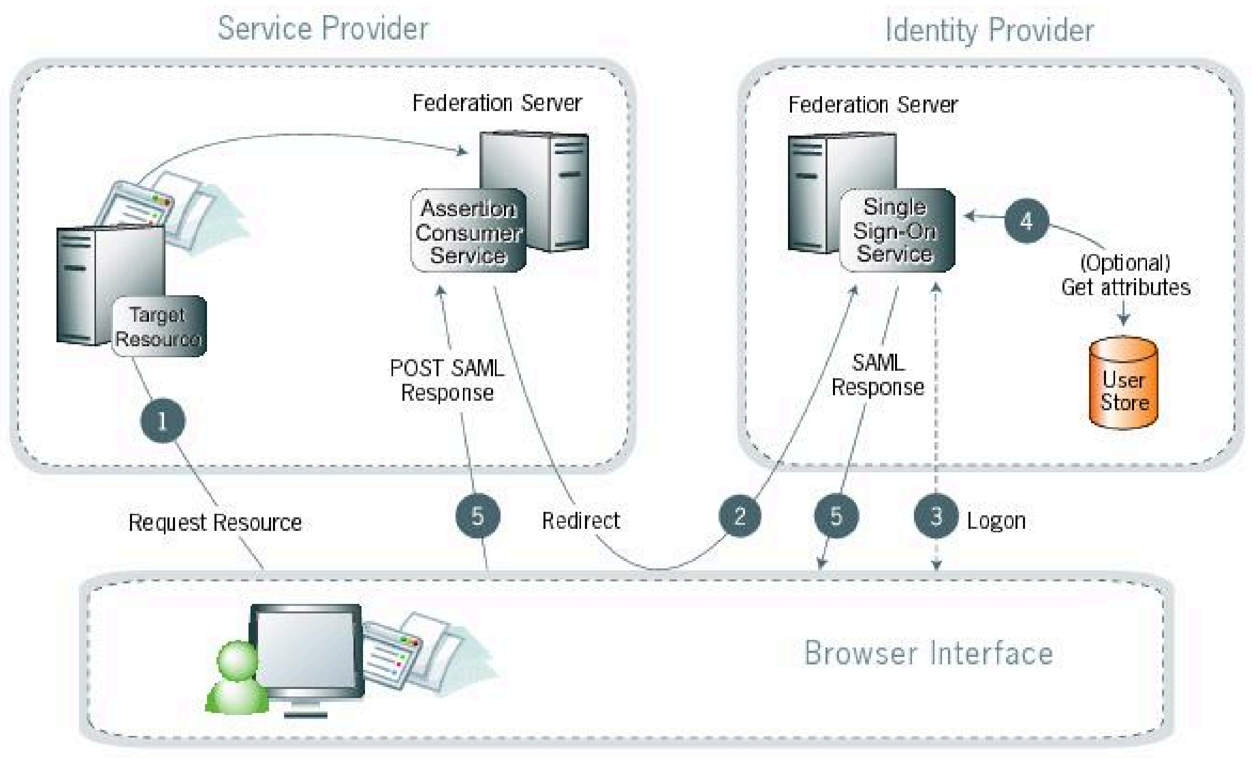
   Note: SAML specifications require that POST responses be digitally signed.

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

About this task

In this scenario, 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.



MY TITLE SP-initiated SSO: redirect/POST

Processing steps:

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 returns an HTTP redirect (code 302 or 303) containing a SAML request for authentication through the user's browser to the IdP's SSO service.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
4. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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.

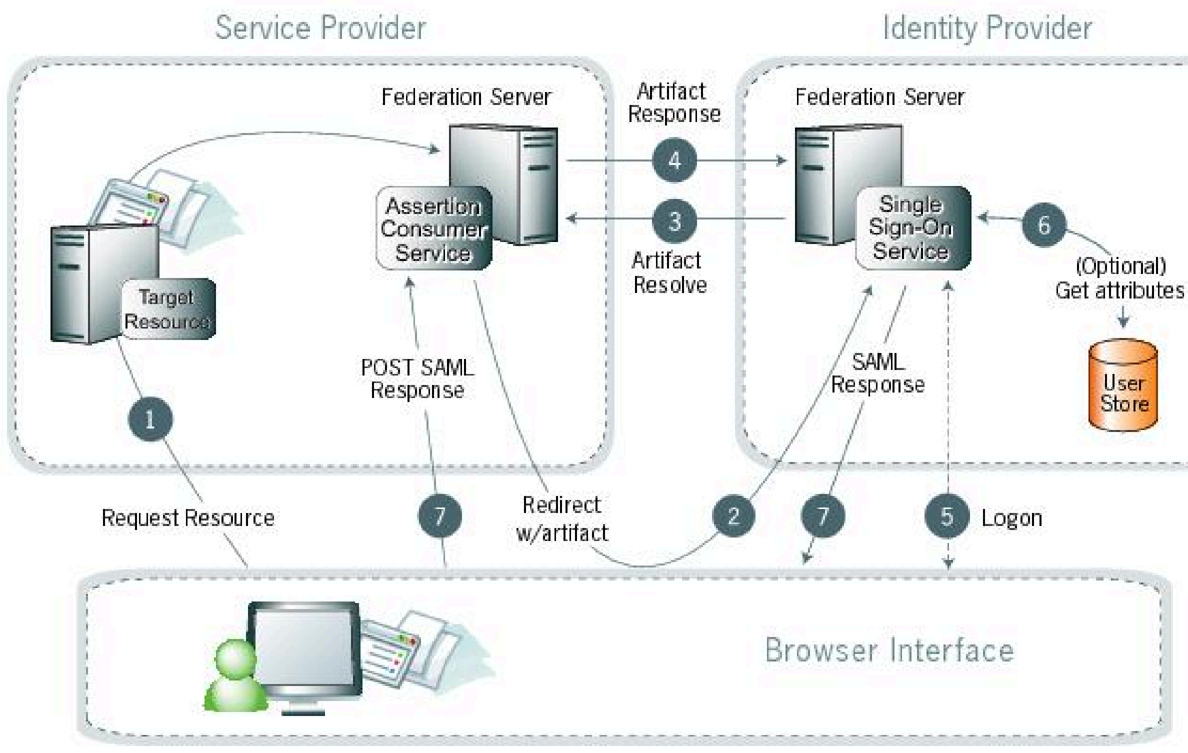
   Note: SAML specifications require that POST responses be digitally signed.

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

About this task

In this scenario, the SP sends a SAML artifact to the IdP via 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.



MY TITLE SP-initiated SSO: artifact/POST

Processing steps:

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 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 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 Artifact Resolution Service (ARS).

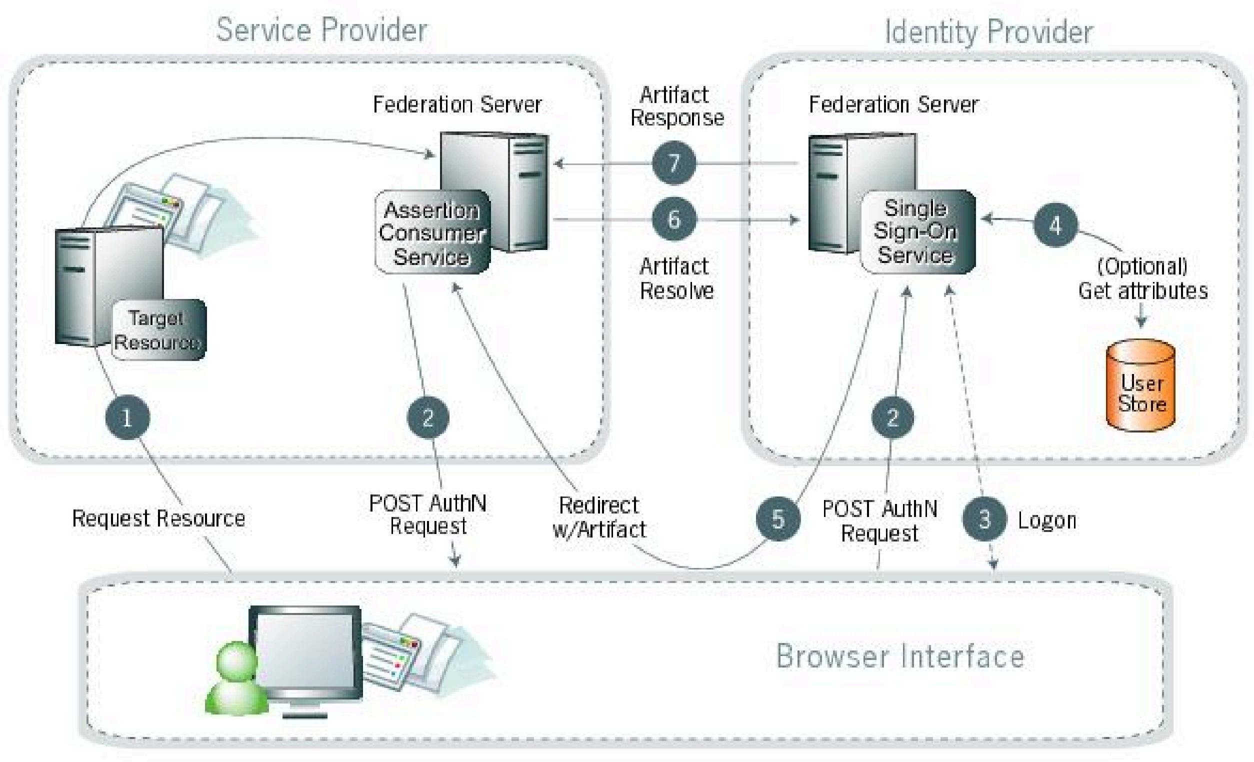
   Note: The SP and IdP's source IDs and remote artifact resolution services are mapped according to the federation agreement made prior to this action.
4. The SP's ARS returns a SAML message containing the previously generated authentication request.
5. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
6. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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 that POST responses be digitally signed.
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

About this task

In this scenario, the SP sends an authentication request to the IdP via HTTP POST. The returned SAML assertion is redirected through the user's browser. The response contains a SAML artifact .



MY TITLE SP-initiated SSO: POST/artifact

Processing steps:

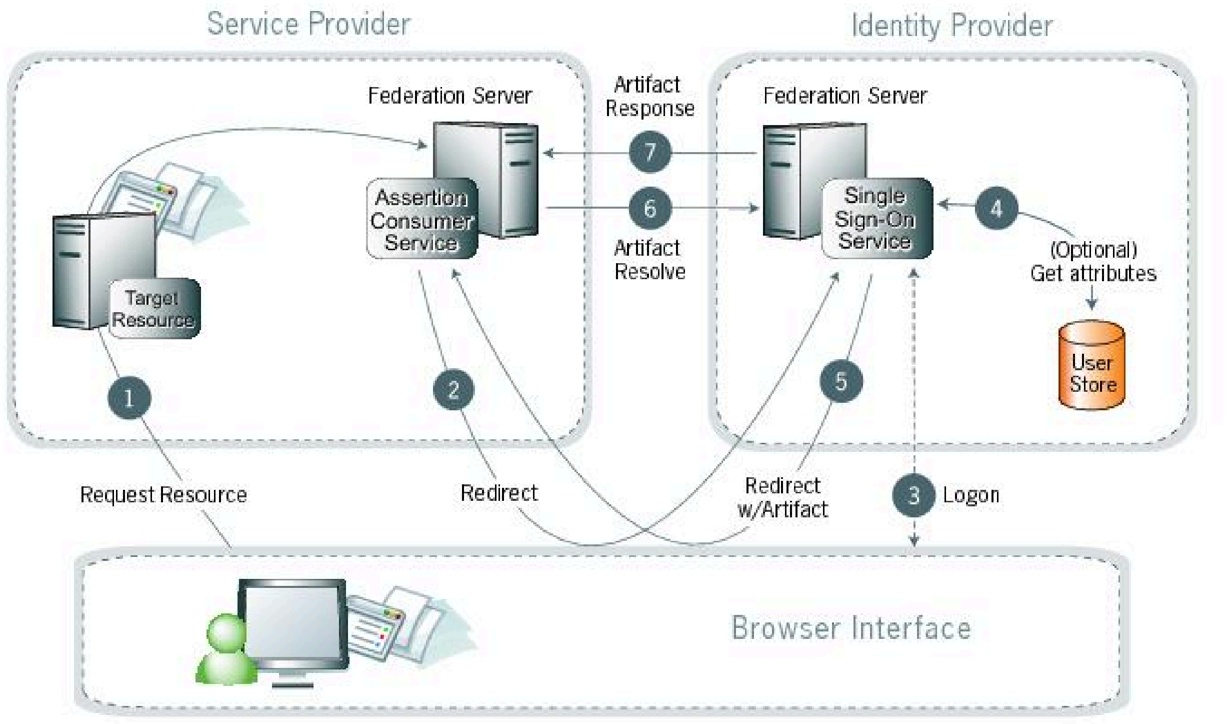
Steps

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 federation server sends an HTML form back to the browser with a SAML request for authentication from the IdP. The HTML form is automatically posted to the IdP's SSO service.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
4. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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 a valid assertion is received, a session is established on the SP and the browser is redirected to the target resource.

SP-initiated SSO—Redirect-Artifact

About this task

In this scenario, the SP sends an HTTP redirect message to the IdP containing a request for authentication. The IdP returns an artifact via HTTP redirect. The SP uses the artifact to obtain the SAML response.



MY TITLE SP-initiated SSO: redirect/artifact

Processing steps:

Steps

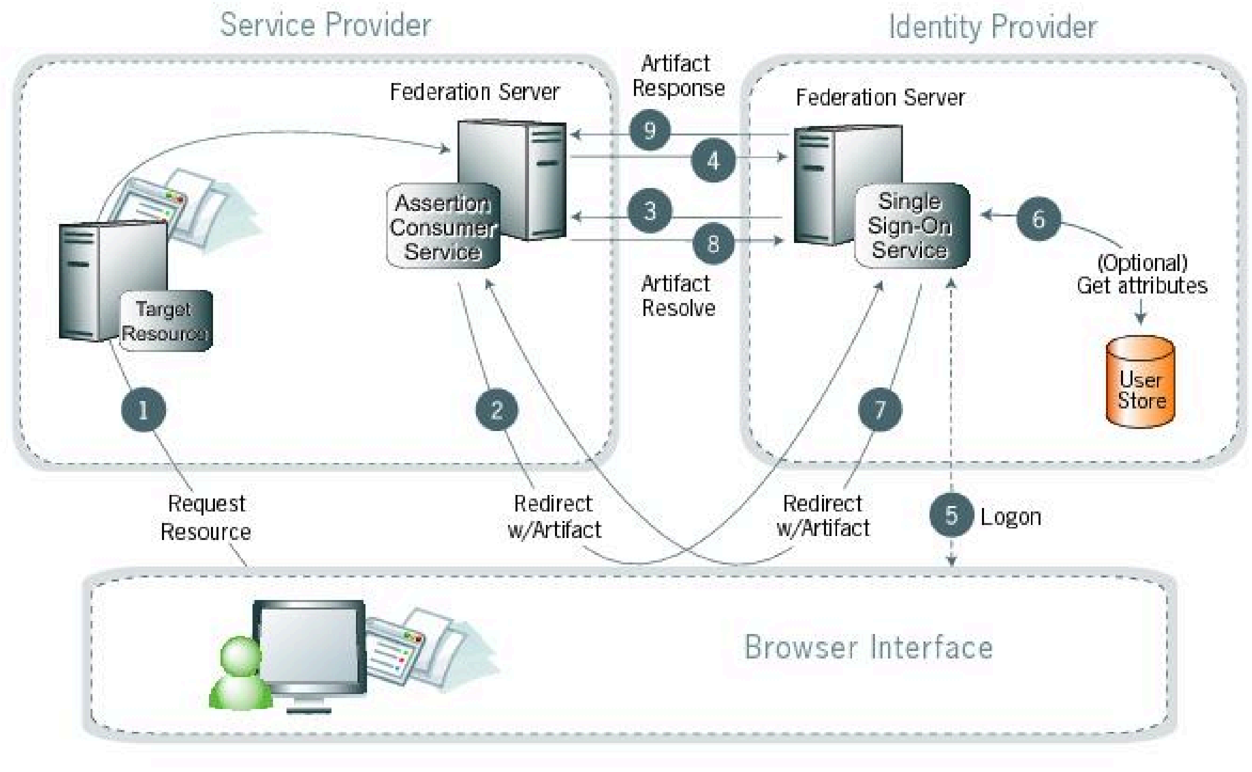
1. A user requests access to a protected SP resource. The user is not logged on to the site. The request is redirected to the federation server to handle authentication.
2. The SP returns an HTTP redirect (code 302 or 303) containing a SAML request for authentication through the user's browser to the IdP's SSO service.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
4. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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 a valid assertion is received, the SP establishes a session and redirects the browser to the target resource.

SP-initiated SSO—Artifact-Artifact

About this task

In this scenario, the SP sends a SAML to the IdP via 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.



MY TITLE SP-initiated SSO: artifact/artifact

Processing steps:

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 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 SSO service.

   Note: The artifact contains the source ID of the SP's artifact resolution service 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.

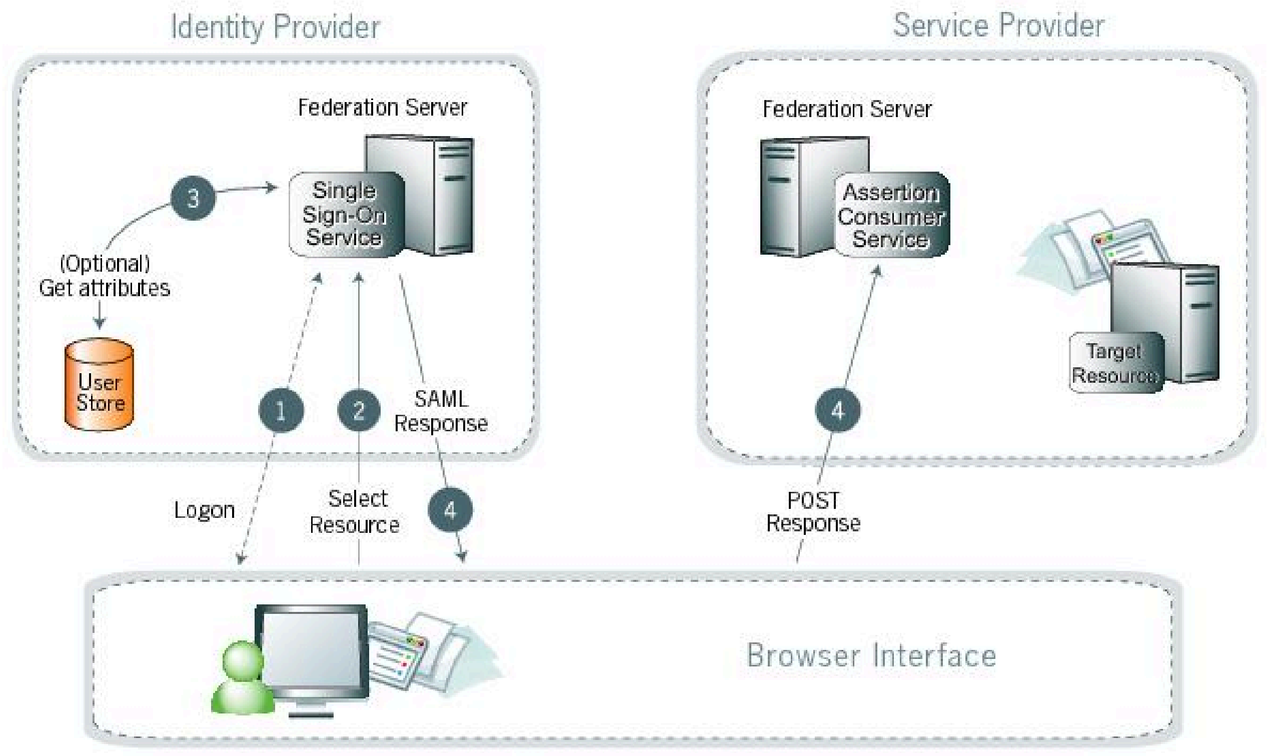
   Note: The SP and IdP's source IDs and remote artifact resolution services are mapped according to the federation agreement prior to this action.

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 if re-authentication is required, the IdP asks for credentials (for example, ID and password) and the user logs on.
6. Additional information about the user may be retrieved from the user data store for inclusion in the SAML response. (These attributes are predetermined as part of the federation agreement between the IdP and the SP—see .)
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 a valid assertion is received, the SP establishes a session and redirects the browser to the target resource.

IdP-initiated SSO—POST

About this task

In this scenario, 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 via HTTP POST.



MY TITLE IdP-initiated SSO: POST

Processing steps:

Steps

1. A user has logged on to the IdP.

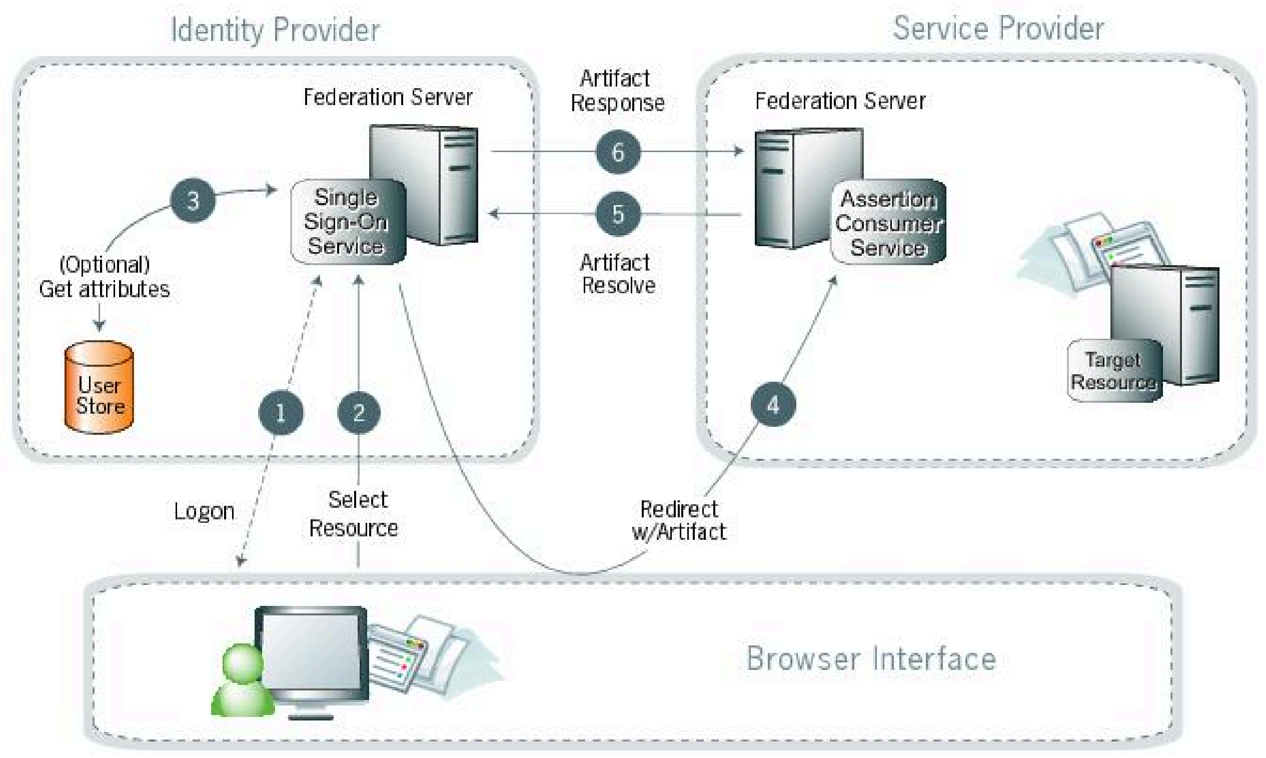
   (If a user has not yet logged on for some reason, he or she is challenged to do so at step 2).
2. The user clicks a link or otherwise requests access to a protected SP resource.
3. Optionally, the IdP retrieves attributes from the user data store.
4. The IdP's SSO service returns an HTML form to the 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 that POST responses be digitally signed.
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

About this task

In this scenario, the IdP sends a SAML artifact to the SP via an HTTP redirect. The SP uses the artifact to obtain the associated SAML response from the IdP.



MY TITLE IdP-initiated SSO: artifact

Processing steps:

Steps

1. A user has logged on to the IdP.

   (If a user has not yet logged on for some reason, he or she is challenged 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 store.
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 a valid assertion is received, the SP establishes a session and redirects the browser to the target resource.

Single logout

The single logout (SLO) profile enables a user to log out of all participating sites in a federated session nearly simultaneously. The user may log out globally from any site, whether SP or IdP, as determined by respective web applications. The associated IdP federation deployment handles all logout requests and responses for participating sites. If a participating site returns an error, other participating sites may not receive their logout requests. In this scenario, PingFederate returns an error message to the end users.

The logout messages may be transported using any combination of bindings described for SSO (POST, artifact, or redirect). Refer to the diagrams under *SAML 2.0 profiles* on page 36 for illustrations of these message flows.

About session cleanup

When an SP receives an SLO request from an IdP, the session creation adapters must handle any session clean-up with respect to the local application.

Attribute Query and XASP

The SAML 2.0 Attribute Query profile allows an SP to request user attributes from an IdP in a secure transaction separate from SSO. The IdP, acting as an *Attribute Authority*, accepts Attribute Queries, performs a data-store 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 (mask) their values in the server and transaction logs.

Web SSO is distinct from the Attribute Query use case; therefore, you can configure PingFederate servers to implement either or both of these profiles without regard to the other.

The X.509 Attribute Sharing Profile (XASP) defines a specialized extension of the general Attribute Query profile. The XASP specification enables organizations with an investment in PKI (Public Key Infrastructure) 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. Once 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 may 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, note that no PingFederate adapter is used in this case.

Standard IdP Discovery

SAML 2.0 IdP Discovery provides a cookie-based look-up mechanism used to identify a user's IdP dynamically during an SP-initiated SSO event, when the IdP is not otherwise specified. This mechanism can be helpful, in particular, in cases where an SP might be a hub for several IdPs in an identity federation.

ⓘ **Tip:** In addition to supporting standard IdP Discovery, PingFederate provides a cross-protocol, proprietary mechanism allowing an 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 *Configure IdP discovery using a persistent cookie* on page 672.

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 may 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.

WS-Federation

About this task

PingFederate supports the WS-Federation Passive Requestor Profile for SP-initiated SSO, enabling interoperability with Microsoft's Active Directory Federation Service (ADFS). This profile provides 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 SLO.

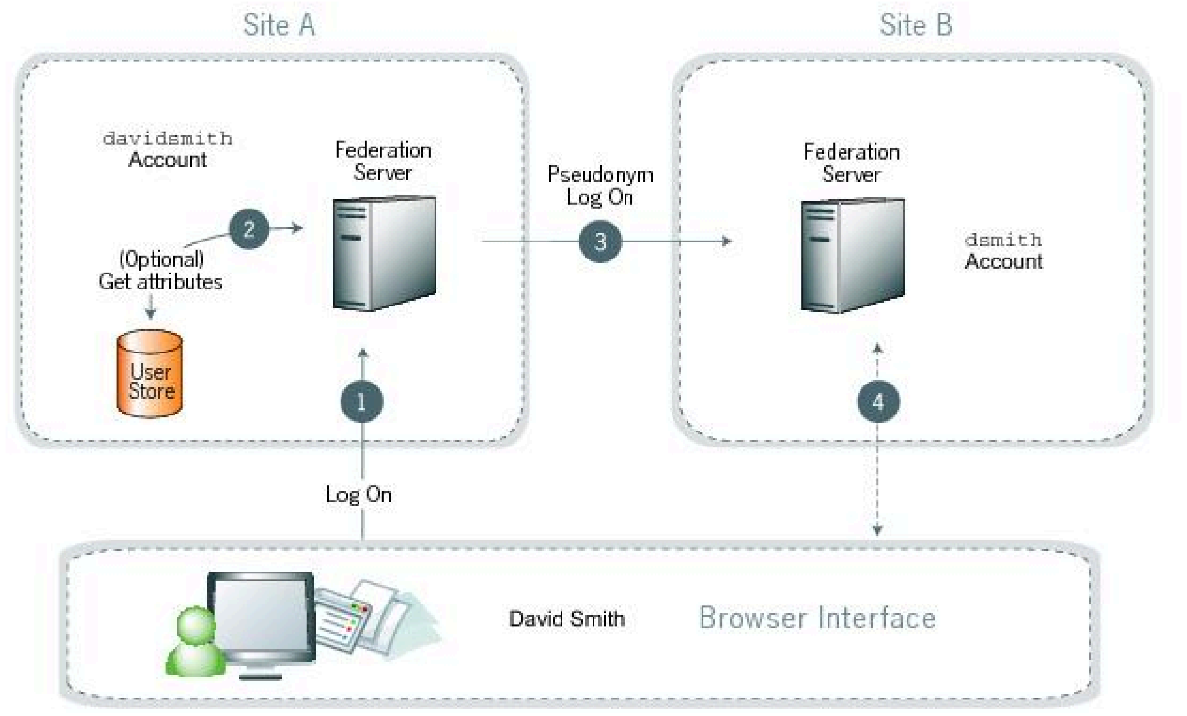
ⓘ **Note:** Unlike SAML, WS-Federation consolidates the endpoints for SLO and SSO. So when you set up a WS-Federation connection in PingFederate, both types of transactions are available to an SP web application that supports them both.

For more information about WS-Federation and the Passive Requestor Profile, see *web services Federation Languages* (docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.html).

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. This method of effectively interconnecting accounts across domains is supported by all protocols.

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



MY TITLE Account linking

Processing steps

1. David Smith logs on to Site A as `davidsmith`. He then decides to access his account on Site B via Site A.
2. Optionally, the federation server looks up additional attributes from the data store.
3. The Site A federation server sends a persistent name identifier (possibly a pseudonym) to Site B, along with any other attributes.

   If a pseudonym is used and other attributes are sent, care must be taken not to send attributes that could be used to identify the subject.
4. The federation server on Site B uses the information to associate the pseudonym with the existing account of dsmith. (Optionally, David is asked to provide consent to the linking.)

   Once the link has been established, it is stored so that David only has to log on to Site A to have access to Site B.

Web services standards

The PingFederate WS-Trust STS is designed to interoperate with many different web-service environments that support varying standards. PingFederate supports multiple versions of SOAP and WS-Trust specifications, and can freely operate with any combinations 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.)

Supported SOAP/WS-Trust versions and corresponding namespaces are listed in following table:

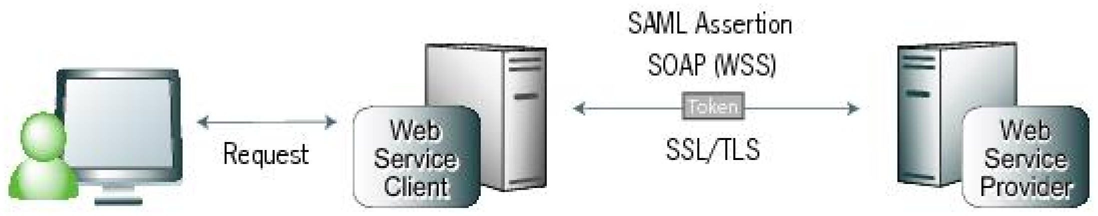
SOAP/WS-Trust versions

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, also WSSE) is a set of specifications defined by the Web Services Security Technical Committee (see *www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss*) at the OASIS standards organization. WSS defines the XML extensions that can be used to secure web service invocations, providing a standard way for partners to add message integrity and confidentiality to their web service interactions. The WSS-defined token profiles describe standard ways of binding security tokens to these messages, enabling a variety of additional capabilities. The WSS technical committee has defined profiles for using SAML assertions, Username, Kerberos, X.509, and other existing security tokens. SSL/ TLS is often used in conjunction with deployments of 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* on page 48).



MY TITLE WSS token transfer

WS-Trust

WS-Trust comprises a protocol for systems and applications to use when requesting a service to issue, validate, and exchange security tokens. Organizations can leverage this protocol to centralize their security-token processing.

The WS-Trust specification also defines the role of a Security Token Service as the entity responsible for responding to requests using the protocol. In this role, the STS creates new security tokens, validates existing security tokens, and/or exchanges security tokens of one type for those of another.

WS-Trust was created by a consortium of leading platform and security vendors who have contributed the protocol to the OASIS standards organization, where it is managed by the WS-SX (Secure Exchange) technical committee (see *www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx*.)

Request types

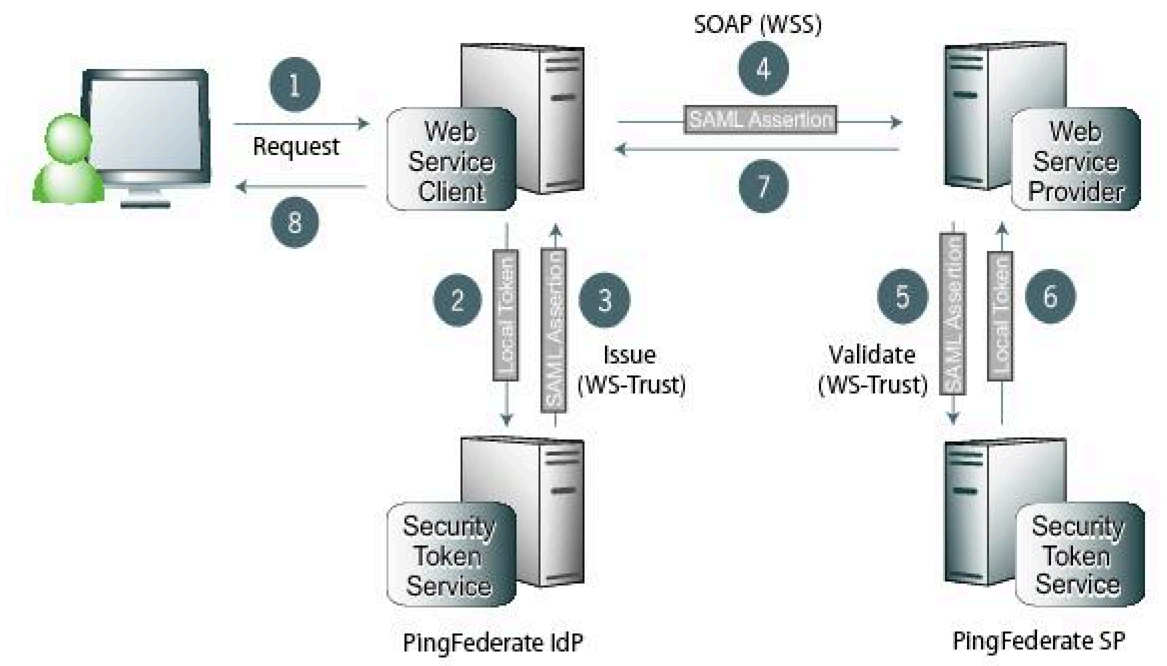
The WS-Trust protocol defines two request types that are particularly useful in securing web services: *Issue* and *Validate*, often associated with the web service client (WSC) and web service provider (WSP), respectively.

- The WSC requests that an 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 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).

The following illustration shows an example of a token exchange, using PingFederate to obtain a SAML assertion to be used in the WSS-secured web service call.

MY TITLE Token exchange (example)



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 WSS header.
5. The WSP retrieves the SAML assertion from the WSS 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 Representational State Transfer (REST) APIs (and non-REST APIs). Rather than 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 and/or policies appropriate to the client and the user. These tokens present less of a security and privacy risk than using secrets (or 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 properly 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. RO is the end user.

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

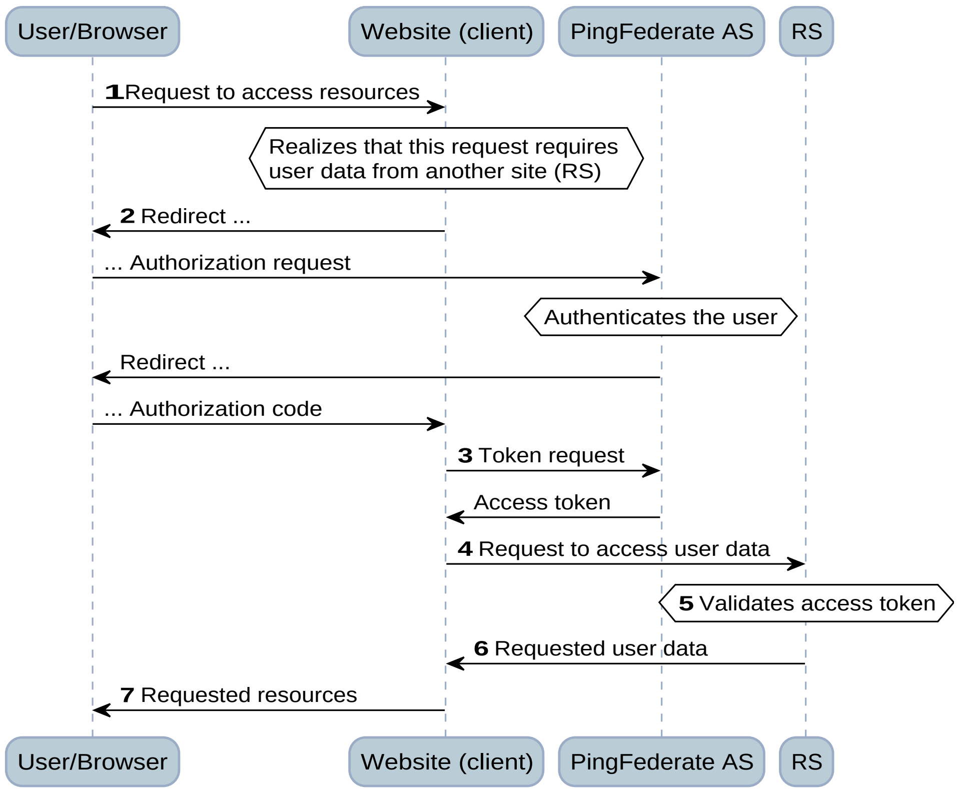
Allows clients to obtain a fresh access token without re-obtaining authorization from the resource owner. It is a long-lived token that a client can trade in to an authorization server to obtain a new (short-lived) access token (with the same attached authorizations as the existing access token).

PingFederate OAuth AS

Based on the Internet Engineering Task Force (IETF) *OAuth 2.0 Authorization Framework* (tools.ietf.org/html/rfc6749), the OAuth AS in PingFederate supports a wide variety of different interaction models appropriate for different types of clients such as a server, a desktop application, or an application on a phone or a tablet. As needed, administrators can also enable cross-origin resource sharing (CORS) support for OAuth endpoints.

Web redirect flow

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 via an HTTP redirect. The client trades the authorization code for an access token, and then uses the token in a API call to obtain data.



MY TITLE Web redirect flow

Processing steps

1. User navigates to an OAuth client website (the requesting site) and requests access to protected resources from another website.

   The OAuth client can optionally include the parameter code_challenge (with or without code_challenge_method) to reduce the risk of code interception attack. For more information, see step 3 and *Proof Key for Code Exchange by (PKCE) OAuth Public Clients* (tools.ietf.org/html/rfc7636).
2. The browser is redirected to the PingFederate OAuth AS with a request for authorization.

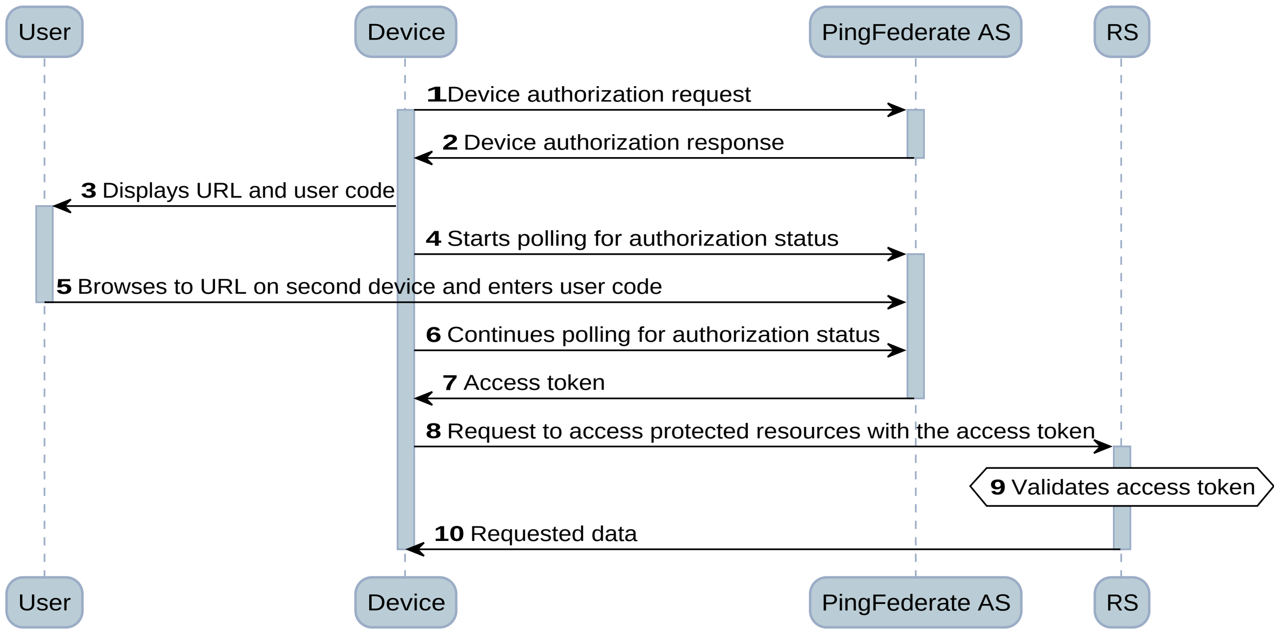
   If the user is not already logged on, the OAuth AS challenges the user to authenticate. The OAuth AS authenticates the user and prompts for authorization. Once the user authorizes, the OAuth AS redirects the browser to the requesting site with an authorization code. If the user does not authenticate, an error is returned rather than the authorization code.
3. The requesting site makes an HTTPS request to the OAuth AS to exchange the authorization code for an access token.

   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 resource server (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

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. For example, 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 smartphone or a computer. For the complete specification, see *OAuth 2.0 Device Authorization Grant* (tools.ietf.org/html/draft-ietf-oauth-device-flow).


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

MY TITLE 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 or without the user code in a query parameter), the user code, and instructions to the user. For 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 must include in its access token request 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:
    a. Go to the user authorization endpoint on a second device that has a browser, such as a smartphone or a computer.
    b. Fulfill the authentication requirements.
    c. Enter the user code or confirm a pre-populated user code.
    d. Approve (or deny) 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 resource server (RS) to access protected resources.
9. The RS validates the access token.
10. The RS provides the requested data to the device.

CIBA grant

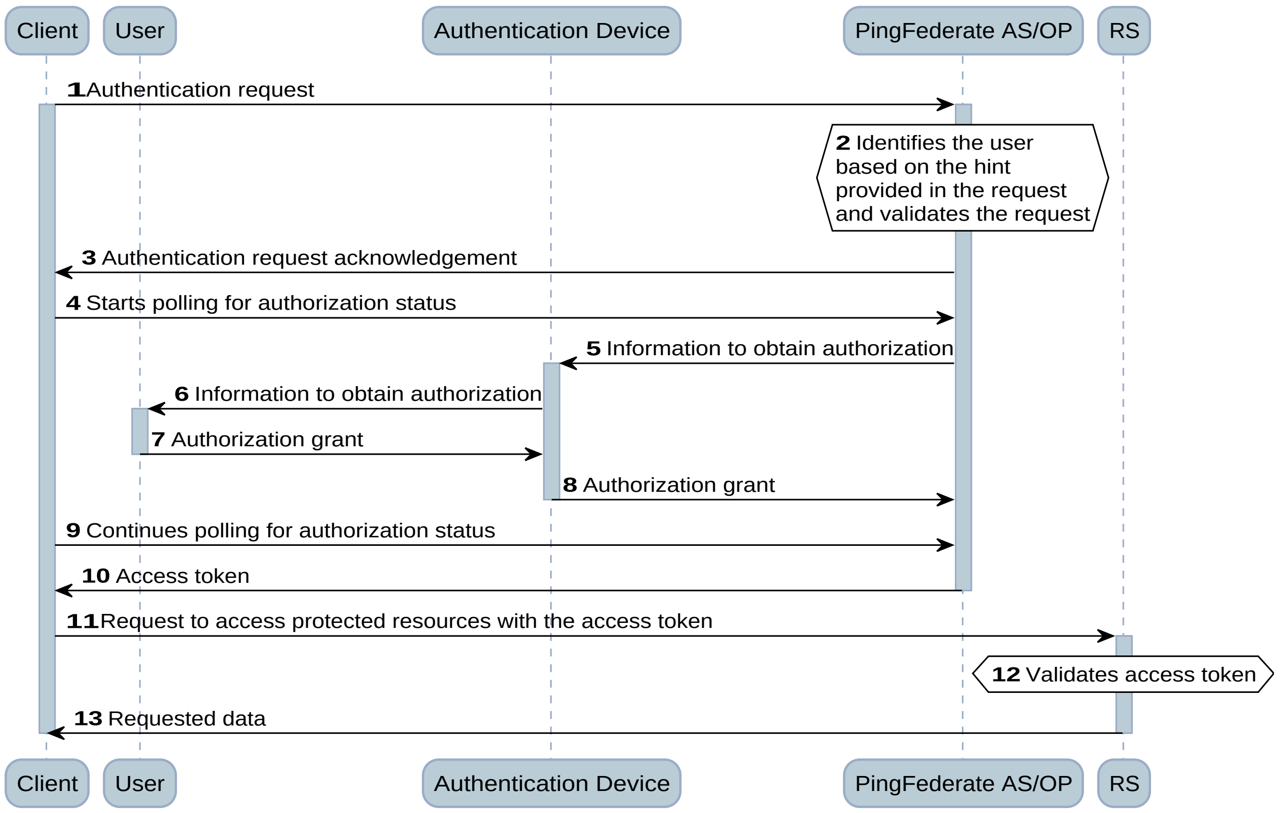
Client Initiated Backchannel Authentication (CIBA) is an extension to OpenID Connect that is gaining interest by organizations that want to improve the end-user experience during authentication and authorization in a federated environment (see *openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html*).

With this extension, user consent can be requested through an out-of-band flow. For example, CIBA improves the user experience when making an online purchase from a merchant as 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.

CIBA allows multiple token delivery methods. PingFederate supports poll and ping. Refer to the subsequent topics for more information about each flow.

CIBA by poll

After receiving an authentication request acknowledgement, the client starts polling the 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 token
13 Requested data

MY TITLE OAuth CIBA grant by poll

Processing steps

1. The client sends an authentication request to PingFederate (the OP) at its client-initiated backchannel authentication (CIBA) endpoint.

   The client must include in its authentication request the requested scope of permissions and one hint for PingFederate to identify the user. When providing an identity hint, the client has three options: login_hint, login_hint_token, or id_token_hint.

   The client may include a user code (user_code) 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. Among other parameters, the response contains an identifier (auth_req_id) 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.

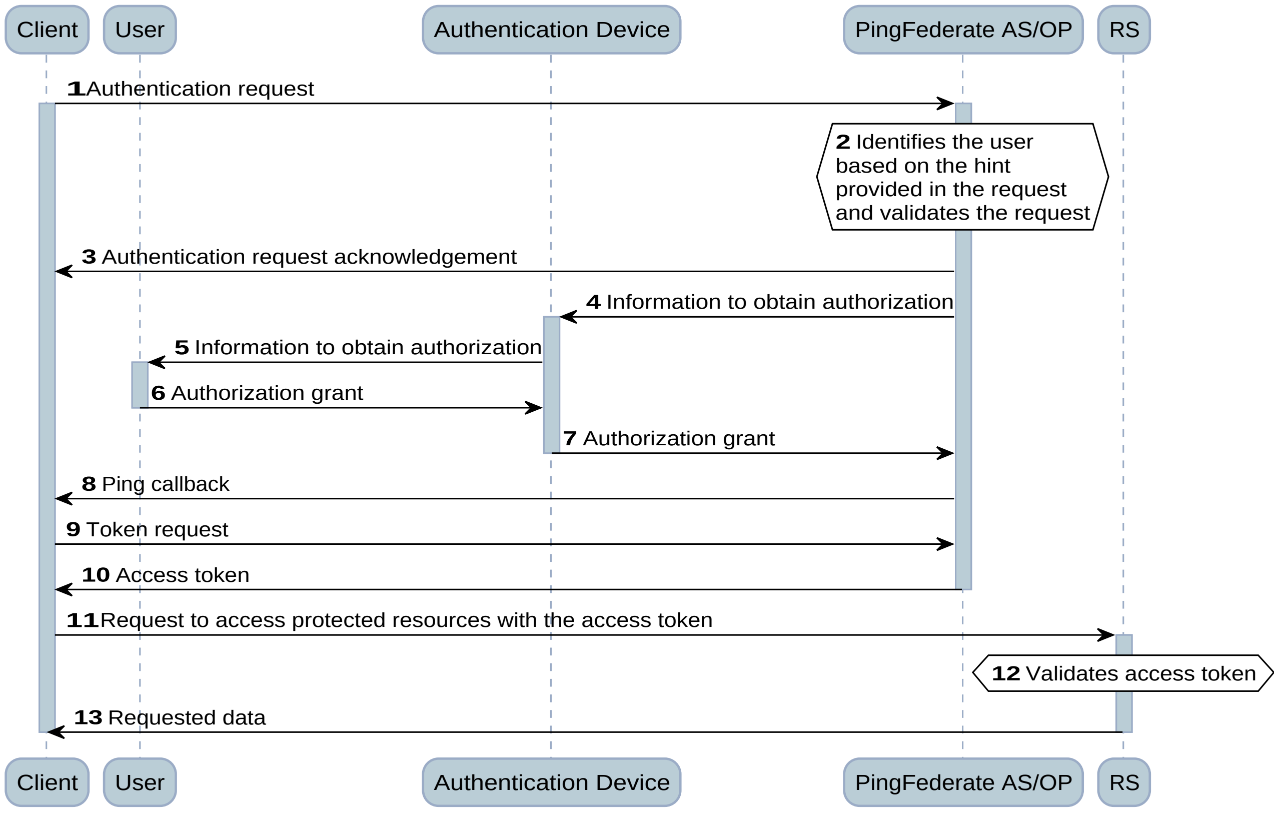
   For each token request it receives, PingFederate returns a token response; the payload varies depending on the authorization status.
5. PingFederate invokes a CIBA authenticator based on the applicable CIBA request policy to reach out to the user with the information (for example, the requested scopes) 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 resource server (RS) to access protected resources.
12. The RS validates the access token.
13. The RS provides the requested data to the client.

CIBA by ping

After receiving an authentication request acknowledgement, the client waits for a ping callback message from the 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 token
13 Requested data

MY TITLE OAuth CIBA grant by ping

Processing steps

1. The client sends an authentication request to PingFederate (the OP) at its client-initiated backchannel authentication (CIBA) endpoint.

   The client must include in its authentication request the requested scope of permissions, one 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, or id_token_hint. For the bearer token, the client must follow the syntax as defined in *RFC 6750, section 2.1* (tools.ietf.org/html/rfc6750#section-2.1) and transmit it by using the client_notification_token parameter.

The client may include a user code (user_code), transmit all request parameters of the authentication request in a signed request object, or do both.

Also per specification and based on mutual agreement, 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 acknowledgement to the client. Among other parameters, the response contains an 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 via the HTTP POST method to the client at its notification endpoint.

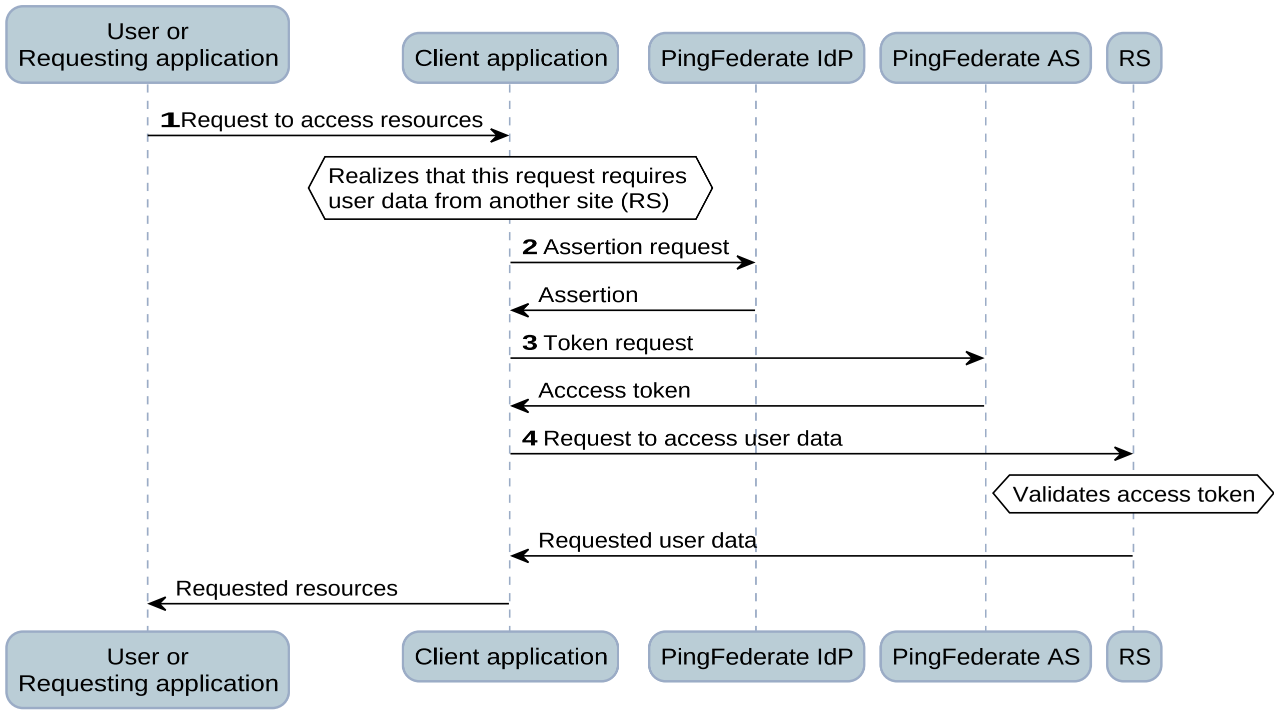
   Per specification, PingFederate includes the client_notification_token value in the Authorization HTTP request header and the auth_req_id value in the message body.
9. The client sends a token request to PingFederate at its token endpoint.

   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 resource server (RS) to access protected resources.
12. The RS validates the access token.
13. The RS provides the requested data to the client.

Assertion grant profile for OAuth 2.0 authorization grants

In this scenario, a client obtains an assertion (a SAML 2.0 bearer assertion or a 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 resource server (RS) to obtain data.



MY TITLE Assertion grant profile

Processing steps

1. Some user-initiated or client-initiated event (for example, 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 Identity Provider (IdP); for example, the PingFederate IdP server.

> **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 SSO profiles here. Refer to the specification for more information.

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.

OpenID Connect support

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 (see *openid.net/connect*).

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

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 (tools.ietf.org/html/rfc7591).

System for Cross-domain Identity Management (SCIM)

PingFederate supports the SCIM 1.1 protocol for outbound and inbound provisioning. At an IdP (outbound) site, you can automatically provision and maintain user accounts at service-provider sites that have implemented SCIM. When PingFederate is configured as an SP (inbound), you can provision and manage user accounts and groups for your own organization automatically using the standard SCIM protocol. The following table provides a brief summary of the supported features.

Feature	Outbound provisioning	Inbound provisioning
SCIM specification	SCIM 1.1	SCIM 1.1
Data format	JSON	JSON
User and group 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	Microsoft Active 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 the website *www.simplecloud.info*.

Transport and message security

The standards generally define two main ways of securing interactions: Secure Sockets Layer with Transport Level Security (SSL/TLS) and digital signatures. SSL/TLS is used in environments where both message confidentiality and integrity are required.

For SAML messaging, digital signatures are used to ensure the identity of both parties involved in the transaction and to validate that a message was received from a particular partner. With PingFederate, you can also choose to encrypt SAML 2.0 messages, including SAML metadata files, as well as WS-Trust STS assertions to achieve increased privacy. For more information, refer to *Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0* (docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf).

Installation

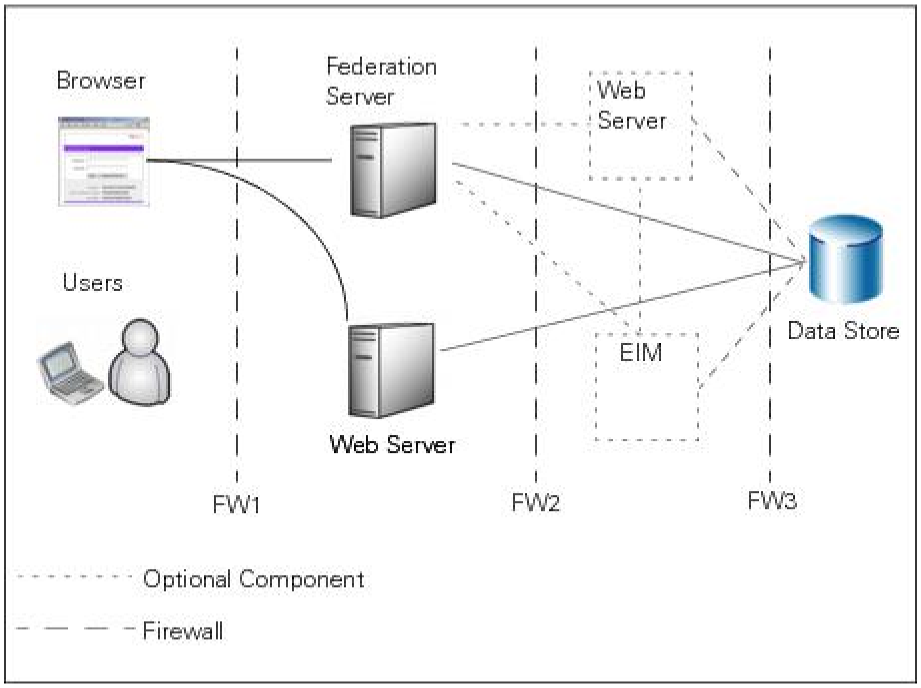
PingFederate is packaged as a standalone server based on J2EE application server technology. A new installation involves the following tasks:

- Determine the deployment architecture.
- Review system and port requirements.
- Install a Java runtime environment.
- Install PingFederate.
- Complete the Initial Setup wizard.

Deployment options

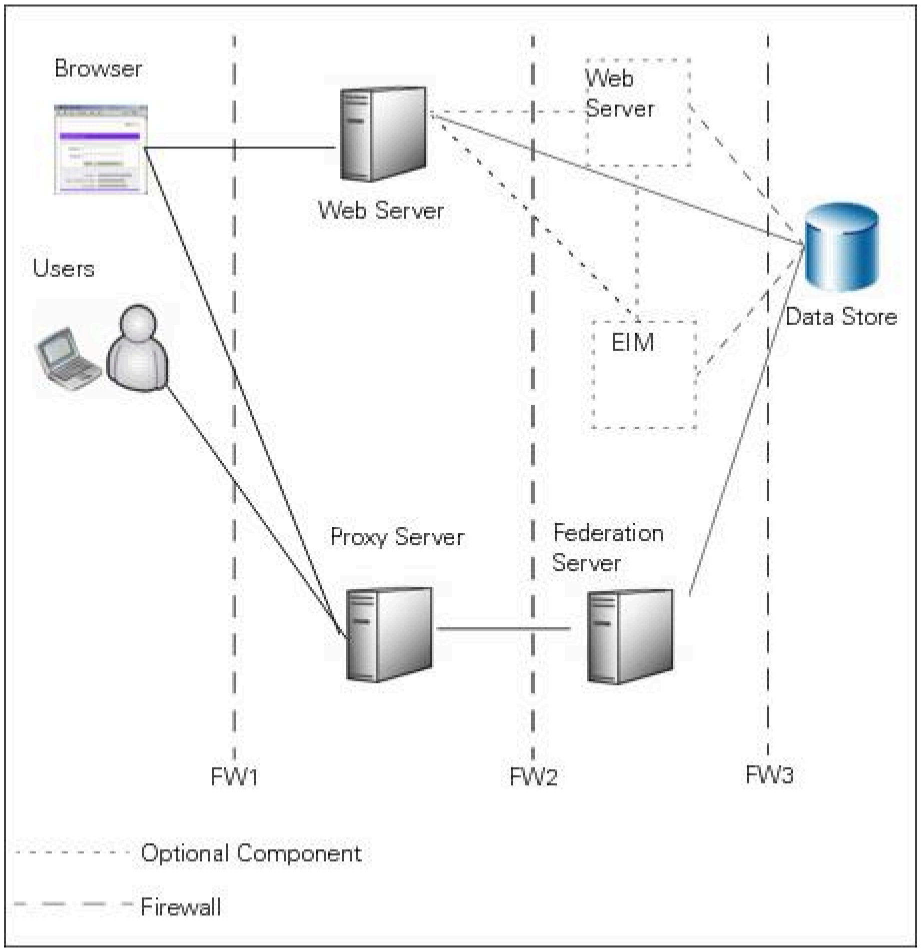
There are many options for deploying PingFederate in your network environment, depending on your needs and infrastructure capabilities.

For example, you can choose a standalone or proxy configuration. The following diagram illustrates PingFederate installed in a demilitarized zone (DMZ):



In this configuration, the users access PingFederate via a web application server, an enterprise identity management (EIM) system, or both. PingFederate may, in turn, retrieve information from a data store 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 the proxy is accessed by users and web browsers. The proxy, in turn, communicates with PingFederate to request SSO.



System requirements

Ping Identity® has qualified the following configurations and certified that they are compatible with the product. Variations of these platforms (for example, differences in operating system version or service pack) are supported up until the point at which an issue is suspected as being caused by the platform or other required software.

Operating systems and virtualization

Note: PingFederate has been tested with default configurations of operating-system components. If your organization has customized implementations or has installed third-party plug-ins, deployment of the PingFederate server may be affected.

Operating systems

- Alpine Linux 3.8.2
- Amazon Linux 2
- Canonical Ubuntu 16.04 LTS
- Canonical Ubuntu 18.04 LTS
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019
- Oracle Enterprise Linux 6.10 (Red Hat compatible kernel)
- Oracle Enterprise Linux 7.6 (Red Hat compatible kernel)
- Red Hat Enterprise Linux ES 6.10

- Red Hat Enterprise Linux ES 7.6
- Red Hat Enterprise Linux ES 8.0
- SUSE Linux Enterprise 12 SP3
- SUSE Linux Enterprise 15

ⓘ **Note:** For Alpine Linux, PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.

Docker support

- Docker version: 18.09.3
- Host operating system: Canonical Ubuntu 18.04 LTS
- Kernel: 4.15.0-1032-aws

Virtualization

Although Ping Identity does not qualify or recommend any specific virtual-machine (VM) or container products other than those listed above, PingFederate has been shown to 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 11
- Amazon Corretto 8
- OpenJDK 11
- Oracle Java SE Development Kit 11 LTS
- Oracle Java SE Runtime Environment (Server JRE) 8

ⓘ **Note:** Ping Identity Java Support Policy applies. Refer to this article for more information.

Browsers

Runtime server

- Apple Safari
- Google Chrome
- Microsoft Edge
- Microsoft Internet Explorer 11
- Mozilla Firefox
- Apple iOS 12 (Safari)
- Google Android 9 (Chrome)

Administrative server

- Google Chrome
- Microsoft Internet Explorer 11
- Mozilla Firefox

TLS protocol

Runtime server and administrative server

- TLS 1.2 and 1.3

> **Note:** TLS 1.3 requires Oracle Java SE Development Kit 11 or OpenJDK 11.

Data store integration

User-attribute lookup

- PingDirectory™ 6.2, 7.0, 7.2, and 7.3
- Microsoft Active Directory 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2016 and 2017
- Oracle Database 12c Release 1
- Oracle MySQL 8.0
- PostgreSQL 9.6.1 and 11.2
- Custom implementation through the PingFederate SDK

SaaS or SCIM outbound provisioning

Provisioning channel data source

- PingDirectory 6.2, 7.0, 7.2, and 7.3
- Microsoft Active Directory 2016
- Oracle Directory Server Enterprise Edition 11g

Provisioning internal data store

- Microsoft SQL Server 2016
- Oracle Database 12c Release 1
- Oracle MySQL 8.0
- PostgreSQL 9.6.1 and 11.2

SCIM inbound provisioning

- Microsoft Active Directory 2016
- Custom implementation through the PingFederate SDK

Just-in-time (JIT) inbound provisioning

- PingDirectory 6.2, 7.0, 7.2, and 7.3
- Microsoft Active Directory 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2016 and 2017

Account linking

- PingDirectory 6.2, 7.0, 7.2, and 7.3
- Microsoft Active Directory 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2016 and 2017
- Oracle Database 12c Release 1

- Oracle MySQL 8.0
- PostgreSQL 9.6.1 and 11.2

OAuth client configuration and persistent grants

- PingDirectory 6.2, 7.0, 7.2, and 7.3
- Microsoft Active Directory 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2016 and 2017
- Oracle Database 12c Release 1
- Oracle MySQL 8.0
- PostgreSQL 9.6.1 and 11.2
- Custom implementation through the PingFederate SDK

Registration and profile management of local identities

- PingDirectory 6.2, 7.0, 7.2, and 7.3

Persistent authentication sessions

- PingDirectory 7.2 and 7.3
- Microsoft SQL Server 2016 and 2017
- Oracle Database 12c Release 1
- Oracle MySQL 8.0
- PostgreSQL 9.6.1 and 11.2

Note: PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see .

Hardware security module (optional)

AWS CloudHSM

- Client software version: 2.0.3

Note: PingFederate must be deployed on one of the Linux operating systems supported by both AWS CloudHSM and PingFederate.

Gemalto SafeNet Luna Network HSM 6

- Appliance software version: 6.3
- Firmware version: 6.27.0
- Client software version: 6.3

Gemalto SafeNet Luna Network HSM 7

- Appliance software version: 7.2.0
- Firmware version: 7.2.0
- Client software version: 7.2.0

nCipher nShield Connect

- Firmware version: 12.40.0
- Client driver version: 12.40.2

ⓘ **Note:** When integrating with a hardware security module (HSM), PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.

Hardware requirements

Minimum hardware recommendations

- Multi-core Intel Xeon processor or higher

  4 CPU/Cores recommended
- 4 GB of RAM

  1.5 GB available to PingFederate
- 1 GB of available hard drive space

ⓘ **Note:** Although it is possible to run PingFederate on less powerful hardware, the following guidelines accommodate disk space for default logging and auditing profiles and CPU resources for a moderate level of concurrent request processing.

Database driver information

PingFederate has been tested with the following vendor-specific JDBC drivers.

Database server	Driver information
Microsoft SQL Server 2016 and 2017	**Driver version information** sqljdbc version 7.2.1 **Driver class** `com.microsoft.sqlserver.jdbc.SQLServerDriver` **JDBC URL** `jdbc:sqlserver://`*`databaseservername`*`;databaseName=`*`databasename`*
Oracle Database 12c Release 1	**Driver version information** ojdbc7 version 12.1.0.2.0 **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 8.0.15 **Driver class** `com.mysql.jdbc.Driver` **JDBC URL** `jdbc:mysql://`*`databaseservername`*`/`*`databasename`*
PostgreSQL 9.6.1 and 11.2	**Driver version information** postgresql version 42.2.5 **Driver class** `org.postgresql.Driver` **JDBC URL** `jdbc:postgresql://`*`databaseservername`*`/`*`databasename`*

For additional information about these drivers, please 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 received by PingFederate from external components. *Outbound* refers to requests sent by PingFederate to external components.

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 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 help. Applicable to the console node in a clustered PingFederate environment.	documentation.pingidentity.com	Used for accessing online help 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 SSO or 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.
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.

Service	Protocol, Direction, Transport, Default Port	Source	Destination	Description
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.
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 the purpose of 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 Directory.
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/siteverify	Used by the HTML Form Adapter when invisible reCAPTCHA from Google is enabled to prevent automated attacks.

ⓘ **Note:** For PingID® integration, refer to *PingID documentation* for a list of required URLs and ports.

Furthermore, additional ports may be required depending on the integration kits deployed and the connecting third-party systems; for example, email server or SMS service provider.

Install Java

About this task

You must install a Java runtime on your server before running PingFederate. PingFederate has been tested in the following Java environments:

- Amazon Corretto 11
- Amazon Corretto 8
- OpenJDK 11
- Oracle Java SE Development Kit 11 LTS
- Oracle Java SE Runtime Environment (Server JRE) 8

ⓘ **Note:** Ping Identity Java Support Policy applies. Refer to this article for more information.

Important: Due to the import restrictions of some countries, Oracle Server JRE (Java SE Runtime Environment) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see the Java 8 release notes from Oracle (www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html).

For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide (docs.oracle.com/en/java/javase/11/migrate/).

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.

   Note: If you intend to use the PingFederate installer for Windows or run PingFederate as a service, you must set the JAVA_HOME variable and modify path variable at the system level; otherwise, you have the options to set the variables at either the system or user level.

Installation options

About this task

You can install PingFederate by running a platform-specific installer (for Microsoft Windows Server or Red Hat Enterprise Linux) or extracting the product distribution ZIP file.

Note: Throughout this documentation, the path to the installation directory, where the `pingfederate` directory is located, is referred to 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 (for example, 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 *Upgrade Guide*.

Install PingFederate on Windows

Steps

1. Request a license key via the Ping Identity licensing website.
2. Ensure you are logged on to your system with appropriate privileges to install and run an application.
3. Verify that the Java runtime is installed and the required environment variables are set correctly (see *Install Java* on page 68).

4. Install PingFederate via the platform-specific installer or the distribution ZIP file.

Installation medium	Steps
Platform-specific installer	Download and run the PingFederate installer for Windows. PingFederate is configured to run as a service and started automatically at the end of the installation process. ⓘ **Note:** The PingFederate installer for Windows is designed to install 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 file. Note that you must manually configure various port settings in the `<pf_install>/pingfederate/bin/run.properties` file (for each instance) to avoid any port conflicts.
Distribution ZIP file	Download and extract the distribution ZIP file into an installation directory.

5. If you have installed PingFederate by the distribution ZIP file, start PingFederate manually by running the following script:

   `<pf_install>/pingfederate/bin/run.bat`

   Wait for the script to finish—the startup process completes when this message appears near the end of the sequence:

   ```
   PingFederate running...
   ```

ⓘ **Tip:** To configure PingFederate to run as a service, follow the steps in *Install PingFederate service on Windows manually* on page 72.

Results

ⓘ **Note:** If your organization requires compliance with FIPS 140-2 or plans on managing keys and certificates using a hardware security module (HSM), see *Supported hardware security modules* on page 90.

Install PingFederate on Red Hat Enterprise Linux

Steps

1. Request a license key via the Ping Identity licensing website.
2. Ensure you are logged on to your system with appropriate privileges to install and run an application.

ⓘ **Note:** You must install and run PingFederate under a local user account.

3. Verify that the Java runtime is installed and the required environment variables are set correctly (see *Install Java* on page 68).

4. Install PingFederate via the platform-specific installer or the distribution ZIP file.

Installation medium	Steps
Platform-specific installer	Download and run the PingFederate install script. PingFederate is configured to run as a service and started automatically at the end of the installation process.
Distribution ZIP file	Download and extract the distribution ZIP file into an installation directory.

5. If you have installed PingFederate by the distribution ZIP file, start PingFederate manually by running the following script:

   `<pf_install>/pingfederate/bin/run.sh`

   Wait for the script to finish—the startup process completes when this message appears near the end of the sequence:

   ```
   PingFederate running...
   ```

> **Tip:** To configure PingFederate to run as a service, follow the steps in *Install PingFederate service on Linux manually* on page 73.

Results

> **Note:** If your organization requires compliance with FIPS 140-2 or plans on managing keys and certificates using a hardware security module (HSM), see *Supported hardware security modules* on page 90.

Install PingFederate on other operating systems

About this task

Refer to *System requirements* on page 60 for a list of qualified Linux operating systems.

Steps

1. Request a license key via the Ping Identity licensing website.
2. Ensure you are logged on to your system with appropriate privileges to install and run an application.

> **Note:** You must install and run PingFederate under a local user account.

3. Verify that the Java runtime is installed and the required environment variables are set correctly (see *Install Java* on page 68).
4. Download and extract the distribution ZIP file into an installation directory (`<pf_install>`).

5. Start PingFederate manually by running the following script:

   `<pf_install>/pingfederate/bin/run.sh`

   Wait for the script to finish—the startup process completes when this message appears near the end of the sequence:

   ```
   PingFederate running...
   ```

   Tip: To configure PingFederate to run as a service, follow the steps in *Install PingFederate service on Linux manually* on page 73.

Results

Note: If your organization requires compliance with FIPS 140-2 or plans on managing keys and certificates using a hardware security module (HSM), see *Supported hardware security modules* on page 90.

Other maintenance tasks

Install PingFederate as a service

About this task

You can set up PingFederate to run in the background as a service on either Windows or Linux.

Tip: If you have installed PingFederate using one of the platform-specific installers, PingFederate has already been configured to run as a service and started automatically at the end of the installation process.

Important: In the event that you want to stop the PingFederate service and start PingFederate manually, you must run the startup script under the same user account that the service uses.

Install PingFederate service on Windows manually

About this task

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 started automatically at the end of the installation process.

Steps

1. Request a license key via the Ping Identity licensing website.
2. Ensure you are logged on to your system with appropriate privileges to install and run an application.
3. Verify that the Java runtime is installed and the required environment variables are set correctly (see *Install Java* on page 68).
4. Download and extract the distribution ZIP file into an installation directory (`<pf_install>`).
5. Start PowerShell or Command Prompt as an Administrator.
6. In PowerShell or Command Prompt, run the `<pf_install>\pingfederate\sbin\win-x86-64\install-service.bat` file to install the service.
7. Open the **Control Panel**# **Administrative Tools**# **Services** management console.

8. Right-click on the **PingFederate** service and select **Start**.

Results

Similar to other services, the PingFederate service is installed and configured to start automatically on reboot.

Install PingFederate service on Linux manually

About this task

If you have installed PingFederate using the install script for Red Hat Enterprise Linux (RHEL), skip these steps because PingFederate has already been configured to run as a service and started automatically at the end of the installation process.

Steps

1. Request a license key via the Ping Identity licensing website.
2. Ensure you are logged on to your system with appropriate privileges to install and run an application.

   > **Note:** You must install and run PingFederate under a local user account.

3. Verify that the Java runtime is installed and the required environment variables are set correctly (see *Install Java* on page 68).
4. Download and extract the distribution ZIP file into an installation directory (`<pf_install>`).
5. Create a new local user account for the PingFederate service; for example, `pingfederate`.

   The service account is referred to as *<pf_user>*.
6. Change the ownership of the PingFederate installation directory (`<pf_install>`) and update the read and write permissions using the following commands:

   ```
   chown -R <pf_user> <pf_install>
   chmod -R 775 <pf_install>
   ```
7. If the operating system supports systemd, follow these steps to install the PingFederate unit file.
   a. Edit the `pingfederate.service` systemd unit file, located in the `<pf_install>/pingfederate/sbin/linux` directory.

      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).

b. Copy the `pingfederate.service` file to the systemd unit files directory; for example, `/etc/systemd/system`.

   The exact location may vary, depending on the operating system. Consult your system administrators, as needed. The rest of the step assumes `/etc/systemd/system` is the systemd unit files directory.

c. Use the following command to update the read and write permissions of the `pingfederate.service` systemd unit file:

```
chmod 664 /etc/systemd/system/pingfederate.service
```

d. Use the following commands to load the new system configuration changes and start the PingFederate service:

```
systemctl daemon-reload ;\
systemctl start pingfederate
```

e. Use the following commands to configure the PingFederate service to start automatically as the server boots.

```
systemctl enable pingfederate ;\
systemctl daemon-reload ;\
systemctl restart pingfederate
```

After setting up the PingFederate systemd unit file, you can use the **`systemctl`** command to manage the PingFederate service.

Sample `systemctl` commands

```
systemctl start pingfederate
systemctl stop pingfederate
systemctl restart pingfederate
systemctl status pingfederate
```

8. If the operating system supports SysV initialization, follow these steps to install the PingFederate script.

a. Edit the `pingfederate` script, located in the `<pf_install>/pingfederate/sbin/linux` directory.

   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 (truncated)

If `<pf_install>` and *<pf_user>* are `/opt/identity.fed` and `pingfederate`, respectively, the required modifications are as follows:

```
...
PF_HOME=/opt/identity.fed/pingfederate
DIR="$PF_HOME/sbin"
USER="pingfederate"
...
```

b. Copy the `pingfederate` script to the SysV initialization directory; for example, `/etc/rc.d/init.d`.

   The exact location may 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.
c. Use the following command to update the read and write permissions of the `pingfederate` SysV initialization script:

```
chmod 755 /etc/rc.d/init.d/pingfederate
```

d. Configure the operating system to start the PingFederate service at various runlevels.

   On an RHEL server, you may use the **Service Configuration** utility to do so.

   Alternatively, you can create symbolic links of the `pingfederate` script in the initialization directories associated with various runlevels manually using the `ln -s` *`source target`* command.

   For example, you may 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
```

   Some operating systems may require a restart of the system to activate the new scripts. Consult your system administrators, as needed.

After setting up the PingFederate SysV initialization script, you can use the **Service Configuration** utility or the **`service`** command to manage the PingFederate service.

Sample `service` commands

```
service pingfederate start
service pingfederate stop
service pingfederate restart
service pingfederate status
```

Uninstall PingFederate

About this task

Uninstalling PingFederate involves removing the PingFederate service (if it was previously installed) and the installation directory (`pf_install`).

Uninstall PingFederate from a Windows server

About this task

Steps

1. Ensure you are logged on to your system with appropriate privileges to uninstall an application.
2. Verify the installation medium in **Control Panel**# **Uninstall a Program**.

   The existence of a **PingFederate** entry indicates that PingFederate was previously installed using the PingFederate installer for Windows; otherwise, it was installed using a distribution ZIP file.
3. Uninstall PingFederate.

Installation medium	Steps
PingFederate installer for Windows	**a.** (Optional) Make a backup copy of the PingFederate installation directory (`<pf_install>`). **b.** Use **Control Panel**# **Uninstall a Program** to uninstall PingFederate, which removes the PingFederate service and the installation directory.
Distribution ZIP file	**a.** Open the **Control Panel**# **Administrative Tools**# **Services** management console. **b.** Right-click on the **PingFederate** service (if found) and select **Stop**, and then run `uninstall-service.bat` from the `<pf_install>\pingfederate\sbin` subdirectory that corresponds to your platform processor. **c.** **Optional:** Remove the PingFederate installation directory (`<pf_install>`).

Uninstall PingFederate from a Linux server

About this task

Steps

1. Ensure you are logged on to your system with appropriate privileges to uninstall an application.
2. Stop and disable the PingFederate service.

   PingFederate systemd service

   Use the following **`systemctl`** commands to stop and disable the PingFederate systemd service:

   ```
   systemctl stop pingfederate ;\
   systemctl disable pingfederate ;\
   systemctl daemon-reload
   ```

   You may also remove the PingFederate systemd unit file (`pingfederate.service`) from the systemd unit files directory (`/etc/systemd/system`) prior to running the `systemctl daemon-reload` command.

   PingFederate SysV initialization script

   On a Red Hat Enterprise Linux (RHEL) server, you may use the **Service Configuration** utility to stop and disable the PingFederate service.

Alternatively, you can stop the service using the **`service`** command (`service pingfederate stop`) and disable the service by removing any symbolic links from various initialization directories.

You may also remove the PingFederate SysV initialization script (`pingfederate`) from the SysV initialization directory (`/etc/rc.d/init.d`).

The exact directory locations may vary, depending on the operating system. Consult your system administrators, as needed.

3. Optional: Remove the PingFederate installation directory (`<pf_install>`).

Start and stop PingFederate

About this task

When you install (or upgrade) PingFederate using its platform-specific installer, PingFederate is configured to run as a service. You can optionally 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.

Depending on the application mode and the operating system, the steps to start, stop, or restart PingFederate vary.

Steps

- Follow the relevant steps to start PingFederate:

Operating system	Application mode	Steps
Windows	Console application	**1.** Open a command prompt. **2.** Go to the `<pf_install>/pingfederate/bin` directory. **3.** Run `run.bat`. **4.** Keep the command prompt open.
	Windows service	**1.** Open the **Control Panel**# **Administrative Tools**# **Services** management console. **2.** Right-click on the PingFederate service and select **Start**. **3.** Close the **Services** management console when the PingFederate Windows service is started.
Linux	Console application	**1.** Open a terminal window. **2.** Go to the `<pf_install>/pingfederate/bin` directory. **3.** Run `run.sh`. **4.** Keep the terminal window open.
	Service	**1.** Open a terminal window. **2.** Enter the system-dependent service command to start the Pingfederate service. **3.** Close the terminal window when the PingFederate service is started.

- Follow the relevant steps to stop PingFederate:

Operating system	Application mode	Steps
Windows	Console application	**1.** Locate the command prompt that is running the PingFederate program. **2.** Use the CTRL+C key combination to terminate the PingFederate program. **3.** Close the command prompt when the PingFederate program is stopped.
	Windows service	**1.** Open the **Control Panel**# **Administrative Tools**# **Services** management console. **2.** Right-click on the PingFederate service and select **Stop**. **3.** Close the **Services** management console when the PingFederate Windows service is stopped.
Linux	Console application	**1.** Locate the terminal window that is running the PingFederate program. **2.** Use the CTRL+C key combination to terminate the PingFederate program. **3.** Close the terminal window when the PingFederate program is stopped.
	Service	**1.** Open a terminal window. **2.** Enter the system-dependent service command to stop the Pingfederate service. **3.** Close the terminal window when the PingFederate service is stopped.

- Follow the relevant steps to restart PingFederate:

Operating system	Application mode	Steps
Windows	Console application	**1.** Locate the command prompt that is running the PingFederate program. **2.** Use the CTRL+C key combination to terminate the PingFederate program. **3.** Run `run.bat` again when the PingFederate program is stopped. **4.** Keep the command prompt open.
	Windows service	**1.** Open the **Control Panel**# **Administrative Tools**# **Services** management console. **2.** Right-click on the PingFederate service and select **Restart**. **3.** Close the **Services** management console when the PingFederate Windows is started.

Operating system	Application mode	Steps
Linux	Console application	**1.** Locate the terminal window that is running the PingFederate program. **2.** Use the CTRL+C key combination to terminate the PingFederate program. **3.** Run `run.sh` again when the PingFederate program is stopped. **4.** Keep the terminal window open.
	Service	**1.** Open a terminal window. **2.** Enter the system-dependent service command to restart the Pingfederate service. **3.** Close the terminal window when the PingFederate service is restarted.

Open PingFederate administrative console

About this task

The PingFederate administrative console is built around a system of wizard-like control screens, in which you configure various settings and components to support your federation use cases.

Steps

1. Start PingFederate.

   In a clustered PingFederate environment, start PingFederate on the console node.
2. Start a web browser.
3. Browse to the following 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 reachable from your computer.

Setup wizard

The first time you access the PingFederate administrative console, it guides you through a series of steps to establish a connection to PingOne® for Enterprise, enable and configure a built-in RADIUS server to integrate PingID® with your VPN, and configure your identity federation settings to deploy a powerful on-premise and cloud-based hybrid SSO and MFA solution.

Alternatively, you may also use the administrative API to complete the setup. For more information, see *PingFederate administrative API* on page 853.

Connect PingFederate to PingOne for Enterprise

About this task

PingOne® for Enterprise is a cloud-based identity as a service (IDaaS) framework for secure identity access management. 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. Select **Yes, Connect to PingOne for Enterprise** and then click **Sign on to PingOne to get your activation key**.
2. Sign on using your PingOne admin portal credentials.

> ⓘ **Tip:** If you do not have an account, you are welcome to register for a free trail.

3. Copy the **Activation Key** value from the PingOne admin portal.
4. Close the browser tab and go back to the PingFederate administrative console.
5. On the **PingOne Account** screen, paste the key value in the **Activation Key** field.
6. Click **Next**.

Results

If you prefer to setup PingFederate without PingOne for Enterprise for now, select **No, Set Up Without PingOne for Enterprise** and then click **Next**. When you are ready to connect PingFederate to PingOne for Enterprise, go to the **System** menu and click **Connect to PingOne for Enterprise**.

Set up with PingOne for Enterprise

Connect to a directory server

About this task

On the **Identities** screen, you can optionally connect to a directory server, which PingFederate can use for PingOne SSO and PingID VPN integration.

Steps

- To enable directory integration, select **Yes, Connect a Directory Server**, provide the required information and then click **Next.**

  For more information about each field, refer to the following table.

Field	Description
Directory Type	Select the type of the directory server from the list. Refer to *System requirements* on page 60 for a list of supported directory servers.
Data Store Name	Enter the name of the data store.
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 may 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 data store queries.

Field	Description
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** screen appears. If you want to continue without a secure connection, click **Next**. Alternatively, you can go back to the **Identities** screen and specify a different directory server.

If the certificate presented by the directory server is not trusted by PingFederate, the **Certificate Error** screen appears. You can import the certificate used by the directory server to establish a secure connection and then click **Next**. Alternatively, you can go back to the **Identities** screen and specify a different directory server.

- 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 POC (proof of concept). Multiple local test accounts are created as a result.

Configure PingOne and PingID options

About this task

Select what you want PingFederate to do.

Steps

1. To enable PingOne® for Enterprise integration, select the **PingOne SSO** check box.

   If you have chosen to connect to a directory server, you may select the **Additional SSO Features** check box to enable provisioning. If the directory is Microsoft Active Directory, selecting this check box also allows you to optionally enable Kerberos authentication. Click **Begin** to configure the applicable settings.
2. To enable and configure a built-in RADIUS server to integrate PingID® with your VPN, select the **PingID VPN (RADIUS)** check box and then click **Begin**.
3. When the administrative console returns you to this screen, click **Next**.

Configure PingOne SSO options
Configure Kerberos authentication

About this task

If you have chosen to connect PingFederate to Microsoft Active Directory, the **Kerberos Authentication** screen becomes available. On the **Kerberos Authentication** screen, you can optionally enable Kerberos authentication for Windows users.

> **Note:** Prior to enabling Kerberos authentication, you must make several Active Directory configuration changes to grant PingFederate Bridge access to the domain and add the domain to PingFederate Bridge. For more information, see *Configuring the Active Directory environment* on page 210.

Steps

1. Select the **Configure Kerberos Authentication** check box and then provide the required information.

   For more information about each field, refer to the following table.

Field	Description
Realm Name	Enter the fully qualified domain name.
Realm Username	Enter the service account that PingFederate can use to communicate with Active Directory for the purpose of Kerberos authentication.
Realm Password	Enter the password associated with the service account.
Internal IP Ranges	Enter one or more network ranges where PingFederate can try authenticating via the Kerberos protocol when handling requests originating from such IP addresses. Typically, these are internal network ranges with access to one or more key distribution centers (KDCs) in your domain. To remove an entry, select it from the list and then click **Delete**.
KDC Hostnames (Optional)	Enter the host name or the IP address of the applicable KDC. This field is optional. Multiple hosts are allowed. If left unspecified, PingFederate uses a DNS query to find a list of KDCs. To remove an entry, select it from the list and then click **Delete**.

2. Optional: Click **Test** to verify your configuration.

   The administrative console returns the test result when the test completes.

> **Note:** When multiple KDCs are returned as a result of a DNS query or provided as part of the configuration, this test stops when it succeeds. As a result, not all KDCs are necessarily verified.

3. Click **Next**.

Results

You must also configure browsers at your site in order to use the Kerberos Adapter to authenticate users. For more information, see *Configure end-user browsers* on page 733.

Configure provisioning to PingOne for Enterprise

About this task

If you have chosen to connect PingFederate to a directory server, the **Provisioning** screen becomes available. On the **Provisioning** screen, you can optionally enable provisioning of users and groups from your directory server to PingOne® for Enterprise. In this configuration, you specify the group where PingFederate should look for member users and provision them to PingOne for Enterprise.

Steps

1. Select the **Configure Provisioning** check box.
2. Enter the distinguished name (DN) of the applicable group in the **Group DN** field.

   The specified group must resides under the hierarchy of the previously defined **Search Base** value (see *Connect to a directory server* on page 80).

3. If you want PingFederate to provision users through nested group membership, select the **Nested** check box

   This check box is not selected by default.
4. Click **Next**.

Review PingOne SSO options

About this task

On the **Summary** screen, review your configuration and perform one of the following tasks.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.
- To discard your changes, click **Cancel**.

Configure PingID VPN (RADIUS) options
Configure basic settings

About this task

On the **Basic Settings** screen, provide the required information to enable the built-in RADIUS server to integrate PingID® with your VPN.

Steps

1. Configure RADIUS client and server information.

   For more information about each field, refer to 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 have chosen to connect PingFederate to a directory server, you can optionally select the **Validate with LDAP** check box to enable first-factor credential validation against that directory server.

   This check box is selected by default.
3. If you have chosen to connect 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.

   For instance, if you have updated the **Search Filter** value from `uid=${username}` to `mail=${username}`, enter `mail` in the **PingID Username Attribute** field.

   (For more information about the **Search Filter** field, see *Connect to a directory server* on page 80.)
4. Click **Next**.

Configure provisioning to PingID

About this task

If you have chosen to connect PingFederate to a directory server, the **Provisioning** screen becomes available. On the **Provisioning** screen, you can optionally enable provisioning of users from your directory server to PingID® . In this configuration, you specify the group where PingFederate should look for member users and update PingID when their email address, first name, or last name has changed. 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.

It is worth noting that this provisioning capability is designed to manage existing PingID accounts. It does not create new PingID users.

Steps

1. Select the **Configure Provisioning** check box.
2. Enter the distinguished name (DN) of the applicable group in the **PingID Group DN** field.

   The specified group must resides under the hierarchy of the previously defined **Search Base** value (see *Connect to a directory server* on page 80).
3. If you want PingFederate to monitor changes for users through nested group membership, select the **Nested** check box

   This check box is not selected by default.
4. Click **Next**.

Review PingID VPN (RADIUS) options

About this task

On the **Summary** screen, review your configuration and perform one of the following tasks.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.
- To discard your changes, click **Cancel**.

Set up without PingOne for Enterprise

Review or import your license

About this task

On the **License** screen, review or import your license.

Steps

1. If a license is installed, the administrative console displays the relevant license information.

   Review your license.

2. If no license is installed or if you wish to import another license file, click **Choose File** to locate and import your license file.

   Alternatively, you may replace the current license with another license file on the **System**# **License Management** screen later.

   ⓘ **Note:** If you do not have a PingFederate license yet, please request a license key at the Ping Identity licensing website or contact sales@pingidentity.com.

3. Click **Next**.

Select your federation roles

About this task

On the **Enable Roles** screen, select the roles of your PingFederate environment.

Steps

1. Select at least one role for your PingFederate environment.
2. Click **Next**.

Configure identity provider settings

About this task

If you have chosen the **Identity Provider** role on the **Enable Roles** screen, the **Identity Provider Configuration** screen becomes available. On the **Identity Provider Configuration** screen, you can connect PingFederate to a directory server. If the directory is Microsoft Active Directory, you can optionally enable Kerberos authentication for Windows users.

Steps

1. Click **Begin** and then follow the on-screen instructions to complete the configuration.
2. When the administrative console returns you to this screen, click **Next**.

Connect to a directory

About this task

On the **Connection** screen, provide the required information to connect to your directory server.

Steps

1. Select a directory type and provide the required information.

   For more information about each field, refer to the following table.

Field	Description
Directory Type	Select the type of the directory server from the list. Refer to *System requirements* on page 60 for a list of supported directory servers.
Data Store Name	Enter the name of the data store.
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 may include a port number.

Field	Description
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 data store 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.

2. Click **Next**.

   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** screen appears. If you want to continue without a secure connection, click **Next**. Alternatively, you can go back to the **Connection** screen and specify a different directory server.

   If the certificate presented by the directory server is not trusted by PingFederate, the **Certificate Error** screen appears. You can import the certificate used by the directory server to establish a secure connection and then click **Next**. Alternatively, you can go back to the **Connection** screen and specify a different directory server.

Configure Kerberos authentication

About this task

If you have chosen to connect PingFederate to Microsoft Active Directory, the **Kerberos Authentication** screen becomes available. On the **Kerberos Authentication** screen, you can optionally enable Kerberos authentication for Windows users.

Important: If you have not created or configured a service account for Kerberos authentication, see *Configuring the Active Directory environment* on page 210 for additional steps. You must have Domain Administrator permissions to make the required changes.

Steps

1. Select the **Configure Kerberos Authentication** check box and then provide the required information.

   For more information about each field, refer to the following table.

Field	Description
Realm Name	Enter the fully qualified domain name.
Realm Username	Enter the service account that PingFederate can use to communicate with Active Directory for the purpose of Kerberos authentication.
Realm Password	Enter the password associated with the service account.

Field	Description
Internal IP Ranges	Enter one or more network ranges where PingFederate can try authenticating via the Kerberos protocol when handling requests originating from such IP addresses. Typically, these are internal network ranges with access to one or more key distribution centers (KDCs) in your domain. To remove an entry, select it from the list and then click **Delete**.
KDC Hostnames (Optional)	Enter the host name or the IP address of the applicable KDC. This field is optional. Multiple hosts are allowed. If left unspecified, PingFederate uses a DNS query to find a list of KDCs. To remove an entry, select it from the list and then click **Delete**.

2. Click **Next**.

Results

Important: Kerberos authentication also requires browser-specific configuration. For more information, see *Configure end-user browsers* on page 733.

Review your identity provider configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.
- To discard your changes, click **Cancel**.

Create an administrator account

About this task

On the **Administrator Account** screen, create an administrative account.

Steps

1. Replace the default value in the **Username** field.

   The default username is `Administrator`.
2. Enter the password associated with this administrator account in the **Password** and **Confirm Password** fields.
3. Click **Next**.

Enter basic information

About this task

On the **Basic Information** screen, enter your federation information.

Steps

1. Verify your **Base URL**. Update as needed.

   The domain portion of the **Base URL** should match the domain name of your organization because it is part of the address where your applications, users, and partners communicate with your PingFederate environment.

   You may also add multiple virtual host names at a later time. For more information, see *Virtual host names* on page 151.
2. If prompted, specify your **Entity ID**.

   This is the unique identifier of your organization. It is how your partners identify you when communicating with you based on the SAML 2.0 specifications.
3. Click **Next**.

Review your configuration

Steps

- On the **Confirmation** screen, review your configuration and perform one of the following tasks:

  Amend your configuration

  Click the corresponding screen title and then follow the configuration wizard to complete the task.

  Keep your configuration

  Click **Next** to apply the configuration and then click **Done**.

  Discard your changes

  Close your browser and restart PingFederate.

PingFederate administrative console

The PingFederate administrative console is built around a system of wizard-like control screens, in which you configure various settings and components to support your federation use cases. The administrative console offers multiple menu choices:

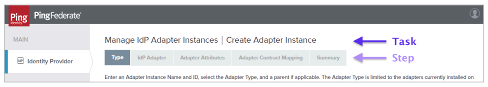
- Identity Provider
- Service Provider
- OAuth Server
- Security
- System

MY TITLE A sample of the Service Provider menu

Note that the menu choices and menu items varies with the federation roles PingFederate plays (see *Choosing roles and protocols* on page 136). Menu items also depend on the permissions assigned to the logged-on administrator (see *Administrative accounts* on page 141).

Tasks and steps

Each broad configuration area is broken down into a series of tasks. Each task consists of a sequence of steps. The tasks and steps appear in the top portion of the screen.



Enter an Adapter Instance Name and ID, select the Adapter Type, and a parent if applicable. The Adapter Type is limited to the adapters currently installed on

MY TITLE Sample tasks and steps

In this example, the primary task is managing one or more IdP adapter instances (**Manage IdP Adapter Instances**). The administrator is working on creating an adapter instance (**Create Adapter Instance**). The current step is about selecting the type the IdP adapter (**Type**). The subsequent steps, which the administrator has not yet reached, are grayed out.

The administrator console displays a summary screen at the end of a task, offering the opportunity to review and make changes as needed.

Some steps provide buttons that branch to dependent tasks with multiple steps. When the dependent tasks are complete, the administrative console brings back the originating tasks for the administrators to continue with the rest of the configuration.

For instance, when creating a connection to a partner, the administrator might need to create a new digital signing certificate, which is a common task with its own set of steps. The connection wizard provides a button to open the task of creating a new signing certificate. When the administrator completes the task, the administrative console brings the administrator back to the connection wizard to finish off the configuration of it.

Note that clicking **Cancel** on any screen discards all new unsaved entries or changes for all steps shown for the current task and returns you to the screen from which you accessed the task.

Console buttons

The navigational and control buttons at the bottom of the administrative console screen change depending on where you are in the configuration process. The following table describes the behavior of these buttons:

Button	Description
Save	Stores information for all steps completed for the current task or any changes made for the current step; returns to the screen from the which the task or step was accessed.This button is available only when the **Save** operation is valid within the current context.
Done	Marks as complete all steps for a current task, but does not save the configuration (because further tasks or steps are necessary); to save entries or changes, click **Save** (or continue the configuration until you see a **Save** button). When creating a new connection, click **Save Draft** (see below).
Save Draft	Stores a new connection configuration for all steps completed up to the current screen in the configuration flow. To return to the draft, click **Manage All** under SP or IdP Connections and then select the draft from the connection list.
Cancel	Returns to the screen from which the current task was accessed; discards any information newly entered or modified for all steps in the task.
Previous	Returns to the previous step (when applicable).
Next	Moves display forward to the next step (when applicable), if all required information is complete in the current step.

CAUTION**:** Do not use the browser's Back, Refresh, or Forward buttons. Instead, always use the navigation buttons, **Previous**, **Next**, or **Done**.

Supported hardware security modules

For optimal security, PingFederate can be configured to use a hardware security module (HSM) for cryptographic material storage and operations. Standards such as the Federal Information Processing Standard (FIPS) 140-2 require the storage and processing of all keys and certificates on a certified cryptographic module.

PingFederate supports:

- AWS CloudHSM
- Gemalto SafeNet Luna Network HSM
- nCipher nShield Connect HSM

Generally speaking, the first step is to install and configure the HSM according to the manufacturer's documentation. Once installed, follow the vendor-specific instructions to configure a new or an existing PingFederate to interact with the HSM for key generation, storage, and operation.

Tip: Starting with PingFederate 8.3, you may enable the HSM hybrid mode, which provides you the choice to store each relevant key and certificate on the HSM or the local trust store. This capability allows your organization to transition the storage of keys and certificates to an HSM without the need to deploy a new PingFederate environment and to mirror the setup. For more information, see *Transition to an HSM* on page 313.

Note: When integrating with a hardware security module (HSM), PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8 because neither Oracle Java SE Development Kit 11 or OpenJDK 11 is supported.

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. It is recommended that you consult your HSM vendor for performance tuning and optimization recommendations if you plan to use an HSM as part of your PingFederate deployment.

Integrate with AWS CloudHSM

About this task

Steps

1. Ensure Oracle Server JRE (Java SE Runtime Environment) 8 is installed on the PingFederate server.

   For more information, see *Install Java* on page 68.

   Additionally, PingFederate must be deployed on one of the Linux operating systems supported by both AWS CloudHSM and PingFederate. For more information, see *System requirements* on page 60 and *Install and Configure the AWS CloudHSM Client (Linux)* in the AWS CloudHSM documentation (docs.aws.amazon.com/cloudhsm/latest/userguide/install-and-configure-client-linux.html).
2. Request a *crypto user* (CU) account from your AWS CloudHSM administrator.

   You need both the username (for example, crypto_user) and the password of this account for your PingFederate installation.

3. **Install and configure your AWS CloudHSM client and tools.**

   Commands vary depending on the operating system. For more information, refer to *Install the AWS CloudHSM Client and Command Line Tools* in the AWS CloudHSM documentation (docs.aws.amazon.com/cloudhsm/latest/userguide/install-and-configure-client-linux.html).
4. **Install AWS CloudHSM software library for Java.**

   Commands vary depending on the Linux operating system. For instructions, refer to *Installing Java Library* in the AWS CloudHSM documentation (docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-install.html#install-java-library).
5. **Validate the AWS CloudHSM client installation.**
   a. Export four environment variables in command line.

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   $ export HSM_PARTITION=PARTITION_1
   $ export HSM_USER=<HSM username>
   $ export HSM_PASSWORD=<password>
   ```
   b. Also in command line, run the following Java program to validate the AWS CloudHSM client installation.

   ```
   $ java -classpath "/opt/cloudhsm/java/*" org.junit.runner.JUnitCore
    TestBasicFunctionality
   ```

   The result should be similar to the following sample output.

   ```
   JUnit version 4.12
   .2019-07-15 19:46:11,129 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:33) - Adding provider.
   2019-07-15 19:46:11,169 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:42) - Logging in.
   2019-07-15 19:46:11,169 INFO  [main] cfm2.LoginManager
    (LoginManager.java:104) - Looking for credentials in
    HsmCredentials.properties
   2019-07-15 19:46:11,170 INFO  [main] cfm2.LoginManager
    (LoginManager.java:122) - Looking for credentials in System.properties
   2019-07-15 19:46:11,170 INFO  [main] cfm2.LoginManager
    (LoginManager.java:130) - Looking for credentials in System.env
       SDK Version: 2.03
   2019-07-15 19:46:11,206 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:54) - Generating AES Key with key size
    256.
   2019-07-15 19:46:11,224 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:63) - Encrypting with AES Key.
   2019-07-15 19:46:11,231 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:84) - Deleting AES Key.
   2019-07-15 19:46:11,232 DEBUG [main] TestBasicFunctionality
    (TestBasicFunctionality.java:92) - Logging out.

   Time: 0.112

   OK (1 test)
   ```

   For more information, refer to *Validating the Installation* in the AWS CloudHSM documentation (docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-install.html#validate-install).
6. **Update the `JAVA_HOME/jre/lib/security/java.security` file in your Java environment and add the `AWSCloudHSMProvider` line to the list of security providers, *immediately after* the `sun.security.provider.Sun` provider; for example:**

   ```
   # List of providers and their preference orders (see above):
   ```

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.pingidentity.crypto.AWSCloudHSMProvider
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=sun.security.ec.SunEC
security.provider.5=com.sun.net.ssl.internal.ssl.Provider
security.provider.6=com.sun.crypto.provider.SunJCE
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
```

7. 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.

8. To enable the Java interface and PingFederate integration, copy the following files to the `JAVA_HOME/jre/lib/ext` directory.
   - `<pf_install>/pingfederate/lib-ext/pf-aws-cloud-hsm-wrapper.jar`
   - `<pf_install>/pingfederate/server/default/lib/disruptor.jar`
   - All files under the `/opt/cloudhsm/java` directory
   - All files under the `/opt/cloudhsm/lib` directory
9. Update the `hivemodule.xml` file.
   a. Edit the `hivemodule.xml` file, located in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.
   b. Look for the **`<!-- Crypto provider -->`** section.
   c. Update the class attribute value of the construct element for both the **`JCEManager`** and **`CertificateService`** service endpoint as follows.

```
...
<!-- Crypto provider -->
<service-point id="JCEManager"
 interface="com.pingidentity.crypto.JCEManager">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.AWSCloudHSMJCEManager"/>
 </invoke-factory>
</service-point>

<service-point id="CertificateService"
 interface="com.pingidentity.crypto.CertificateService">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.AWSCloudHSMCertificateServiceImpl"/>
 </invoke-factory>
</service-point>
...
```

10. Update the `<pf_install>/pingfederate/bin/run.properties` file.
   a. Change the value of the pf.hsm.mode property from `OFF` to `AWSCLOUDHSM`.
   b. If you are setting up a new PingFederate installation, set the value of the pf.hsm.hybrid property to `false`. When set to `false`, as you create or import certificates (such as your signing certificate or your encryption key), the certificates are stored on your HSM.

      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 *Transition to an HSM* on page 313.

11. From the `<pf_install>/pingfederate/bin` directory, run the `hsmpass.sh` script for Linux.

    Enter the password for the CU account when prompted (see *step 2*).

    This procedure sets and securely stores the password for communication to the HSM from PingFederate.

12. If the username of the CU account is *not* crypto_user, update the `com.pingidentity.crypto.AWSCloudHSM.xml` file.

    a. Edit the `com.pingidentity.crypto.AWSCloudHSM.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

       The unmodified version of the `com.pingidentity.crypto.AWSCloudHSM.xml` file reads:

```
<?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>
```

    b. Replace `crypto_user` with the username of the CU account.

       For example, if the username of the CU account is example_user, update as follows.

```
<?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>
```

13. If you are setting up a new or configuring an existing PingFederate cluster, repeat these steps on each node.

Results

This completes the steps required to configure PingFederate for use with AWS CloudHSM. You may start the new PingFederate server or restart the existing PingFederate server.

AWS CloudHSM operational notes

Some restrictions apply to PingFederate operations when using an HSM:

- PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.
- PingFederate does not store public certificates (for the purposes of signature verification, encryption, and back-channel authentication) on the hardware module. In this case, certificates are stored in the local trust store located on the file system.
- As an OpenID Provider, PingFederate can use static or dynamically rotating keys to sign ID tokens, JWTs for client authentication, and OpenID Connect request objects. When dynamically rotating keys are used (the default configuration), the short-term keys are stored in memory, not on the HSM. (If static keys are used, they can be stored on HSM.)
- Private keys are not exportable. When configured for use with the HSM, administrative-console options for this feature are disabled. 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, any deletion or creation of objects on the HSM not executed via the PingFederate user interface will not be recognized or operational.

  For example, during the course of normal PingFederate operation 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 then delete object C and attempt to recover it via the data archive, PingFederate fails, producing various exceptions. Because the data archive contains a reference to the object and the object has been deleted from the HSM, it is not possible to use that data archive again.

- Not all cipher suites in a standard Java configuration are available. They are limited to those listed in the `com.pingidentity.crypto.AWSCloudHSMJCEManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Integrate with Gemalto SafeNet Luna Network HSM

About this task

Steps

1. Ensure Oracle Server JRE (Java SE Runtime Environment) 8 is installed on the PingFederate server.

   To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see *Install Java* on page 68.
2. Install and configure your Gemalto SafeNet Luna Network HSM, including the optional package for Java (referred to as the JSP), according to SafeNet's instructions.

   This includes the creation of a partition, creation of a Network Trust Link (NTL), and assignment of a client to a partition. Ensure that you can perform the **`vtl verify`** command indicating that you are communicating securely and properly to the HSM.

   Delete any unnecessary keys or objects that may have been created while testing communication to the HSM from the host that runs PingFederate.

   Note the password used to open communication to the HSM via the NTL. You need this for your PingFederate installation.
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 `LUNA_HOME\jsp\lib\LunaAPI.dll` file to an arbitrary directory and add the directory's path as a system variable. Alternatively, you can copy the file to the Windows system directory (`C:\Windows\System32`). Copy the `LUNA_HOME\jsp\lib\LunaProvider.jar` file to the `JAVA_HOME\jre\lib\ext` directory.
Linux	Copy the `libLunaAPI.so` and `LunaProvider.jar` files from the `LUNA_HOME/jsp/lib` directory to the `JAVA_HOME/jre/lib/ext` directory.

   SafeNet provides some sample Java applications that may be run to ensure that the Java HSM interface is working properly prior to installing PingFederate. Please refer to the HSM documentation from Gemalto for more information.
4. Update the `JAVA_HOME/jre/lib/security/java.security` file in your Java environment and add the `LunaProvider` line to the list of security providers, *immediately before* the `sun.security.ec.SunEC` provider; for example:

```
# List of providers and their preference orders (see above):
security.provider.1=sun.security.provider.Sun
security.provider.2=sun.security.rsa.SunRsaSign
security.provider.3=com.safenetinc.luna.provider.LunaProvider
security.provider.4=sun.security.ec.SunEC
security.provider.5=com.sun.net.ssl.internal.ssl.Provider
security.provider.6=com.sun.crypto.provider.SunJCE
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
```

5. 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.

6. Update the `hivemodule.xml` file.
   a. Edit the `hivemodule.xml` file, located in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.
   b. Look for the **`<!-- Crypto provider -->`** section.
   c. Update the class attribute value of the construct element for both the **`JCEManager`** and **`CertificateService`** service endpoint as follows.

```
...
<!-- Crypto provider -->
<service-point id="JCEManager"
 interface="com.pingidentity.crypto.JCEManager">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.LunaJCEManager"/>
 </invoke-factory>
</service-point>

<service-point id="CertificateService"
 interface="com.pingidentity.crypto.CertificateService">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.LunaCertificateServiceImpl"/>
 </invoke-factory>
</service-point>
...
```

7. Update the `<pf_install>/pingfederate/bin/run.properties` file.
   a. Change the value of the pf.hsm.mode property from `OFF` to `LUNA`.
   b. If you are setting up a new PingFederate installation, set the value of the pf.hsm.hybrid property to `false`. When set to `false`, as you create or import certificates (such as your signing certificate or your encryption key), the certificates are stored on your HSM.

      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 *Transition to an HSM* on page 313.

8. From the `<pf_install>/pingfederate/bin` directory, run the `hsmpass.bat` batch file for Windows or the `hsmpass.sh` script for Linux.

   Enter the NTL password when prompted (see *step 2*).

   This procedure sets and securely stores the password for NTL communication to the HSM from PingFederate.

> ⓘ **Note:** The Gemalto SafeNet Luna Network HSM may be configured in a high-availability group. To do so, please refer to the SafeNet distributed-installation instructions. To properly synchronize data, ensure that the HAOnly property is enabled using this command:
>
> **`vtl haAdmin –HAOnly –enable`**

9. If you are setting up a new or configuring an existing PingFederate cluster, repeat these steps on each node.

Results

This completes the steps required to configure PingFederate for use with Gemalto SafeNet Luna Network HSM. You may start the new PingFederate server or restart the existing PingFederate server.

Important: To ensure expected behavior, SafeNet recommends restarting dependent processes such as PingFederate (including all server nodes in a cluster) whenever the Luna HSM is restarted.

SafeNet Luna Network HSM operational notes

Some restrictions apply to PingFederate operations when using an HSM:

- PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.
- PingFederate does not store public certificates (for the purposes of signature verification, encryption, and back-channel authentication) on the hardware module. In this case, certificates are stored in the local trust store located on the file system.
- As an OpenID Provider, PingFederate can use static or dynamically rotating keys to sign ID tokens, JWTs for client authentication, and OpenID Connect request objects. When dynamically rotating keys are used (the default configuration), the short-term keys are stored in memory, not on the HSM. (If static keys are used, they can be stored on HSM.)
- Private keys are not exportable. When configured for use with the HSM, administrative-console options for this feature are disabled. Only the public portion of generated keys is exportable.
- When running in FIPS 140-2 level 3 compliance (also known as strict FIPS mode) private keys can not be imported. In this case 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, any deletion or creation of objects on the HSM not executed via the PingFederate user interface will not be recognized or operational.

  For example, during the course of normal PingFederate operation 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 then delete object C and attempt to recover it via the data archive, PingFederate fails, producing various exceptions. Because the data archive contains a reference to the object and the object has been deleted from the HSM, it is not possible to use that data archive again.
- Not all cipher suites in a standard Java configuration are available. They are limited to those listed in the `com.pingidentity.crypto.LunaJCEManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Integrate with nCipher nShield Connect HSM

About this task

Steps

1. Ensure Oracle Server JRE (Java SE Runtime Environment) 8 is installed on the PingFederate server.

   To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see *Install Java* on page 68.
2. Install and configure your nCipher 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 your installation, refer to the HSM documentation from nCipher to see how to make your PingFederate server a client of an HSM server.

   Note: PingFederate only supports Operator Card Set (OCS) protected keys. Note the password used for the OCS; you will need the password for your installation of PingFederate.

4. To enable the Java interface, copy the `NFAST_HOME/java/classes/nCipherKM.jar` file to the `JAVA_HOME/jre/lib/ext` directory.

   nCipher provides some sample Java applications that may be run to ensure that the Java HSM interface is working properly prior to installing PingFederate. Please refer to the HSM documentation from nCipher for more information.

5. Update the `JAVA_HOME/jre/lib/security/java.security` file in your Java environment and add the `nCipherKM` line to the list of security providers, *after* all the `sun` providers; for example:

```
# List of providers and their preference orders (see above):
security.provider.1=sun.security.provider.Sun
security.provider.3=com.sun.net.ssl.internal.ssl.Provider
security.provider.4=com.sun.crypto.provider.SunJCE
security.provider.5=sun.security.jgss.SunProvider
security.provider.6=com.sun.security.sasl.Provider
security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.8=sun.security.smartcardio.SunPCSC
security.provider.9=sun.security.mscapi.SunMSCAPI
security.provider.10=com.ncipher.provider.km.nCipherKM
```

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. Update the `hivemodule.xml` file.
   a. Edit the `hivemodule.xml` file, located in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.
   b. Look for the **`<!-- Crypto provider -->`** section.
   c. Update the class attribute value of the construct element for both the **`JCEManager`** and **`CertificateService`** service endpoint as follows.

```
...
<!-- Crypto provider -->
<service-point id="JCEManager"
 interface="com.pingidentity.crypto.JCEManager">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.NcipherJCEManager"/>
 </invoke-factory>
</service-point>

<service-point id="CertificateService"
 interface="com.pingidentity.crypto.CertificateService">
 <invoke-factory>
  ...
  <construct class="com.pingidentity.crypto.NcipherCertificateServiceImpl"/>
 </invoke-factory>
</service-point>
...
```

8. Update the `<pf_install>/pingfederate/bin/run.properties` file.
    a. Change the value of the pf.hsm.mode property from `OFF` to `NCIPHER`.
    b. If you are setting up a new PingFederate installation, set the value of the pf.hsm.hybrid property to `false`. When set to `false`, as you create or import certificates (such as your signing certificate or your encryption key), the certificates are stored on your HSM.

        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 *Transition to an HSM* on page 313.
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 *step 3*).

    This procedure sets and securely stores the password for communication to the HSM from PingFederate.
10. 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.

    a. On the console node, locate the `<pf_install>/pingfederate/server/default/data` directory and create a sub directory named `ncipher-kmdata-local`.
    b. 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`.
    c. Create a new environment variable named NFAST_KMLOCAL and set it to `<pf_install>/pingfederate/server/default/data/ncipher-kmdata-local`

        Note: You must perform define this environment variable on all servers within the cluster.
    d. Restart the nShield Connect hardserver on all PingFederate servers in the cluster. (See the HSM documentation from nCipher for instructions on restarting the hardserver.)
    e. Log on to the PingFederate administrative console, go to the **System# Cluster Management** screen and then click **Replicate Configuration** to push the configuration changes, which includes the nShield data, to the engine nodes.

Results

This completes the steps required to configure PingFederate for use with nCipher nShield Connect HSM. You may start the new PingFederate server or restart the existing PingFederate server.

Important: To ensure expected behavior, PingFederate (including all server nodes in a cluster) should be restarted whenever the nShield HSM is restarted.

nShield Connect HSM operational notes

Some restrictions apply to PingFederate operations when using an HSM:

- PingFederate must be deployed with Oracle Server JRE (Java SE Runtime Environment) 8.
- When PingFederate is integrated with nCipher nShield Connect on a platform with Oracle Server JRE (Java SE Runtime Environment) 8u102, runtime errors may occur when handling certificates with a signing algorithm of RSA SHA256, SHA384, or SHA512. Upgrading to Oracle Server JRE (Java SE Runtime Environment) 8u112 resolves these runtime errors.

- PingFederate only supports Operator Card Set (OCS) protected keys. If you use a standard (non-persistent) OCS, the HSM removes the protected keys from its memory when the card is removed from the smart card reader. Requests will likely fail because almost all requests require cryptographic processing. To resume operations, you must insert the card back to the smart card reader and then restart PingFederate.

  Alternatively, you may 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. Note that no new keys and certificates can be created and stored on the HSM until the card is inserted back to the HSM. (No restart of PingFederate is required.) For more information about persistent OCS, please consult your HSM vendor.
- As an OpenID Provider, PingFederate can use static or dynamically rotating keys to sign ID tokens, JWTs for client authentication, and OpenID Connect request objects. When dynamically rotating keys are used (the default configuration), the short-term keys are stored in memory, not on the HSM. (If static keys are used, they can be stored on HSM.)
- Private keys are not exportable. When configured for use with the HSM, administrative-console options for this feature are disabled. Only the public portion of generated keys is exportable.
- When running in FIPS 140-2 level 3 compliance (also known as strict FIPS mode) private keys can not be imported. In this case 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, any deletion or creation of objects on the HSM not executed via the PingFederate user interface will not be recognized or operational.

  For example, during the course of normal PingFederate operation 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 then delete object C and attempt to recover it via the data archive, PingFederate fails, producing various exceptions. Because the data archive contains a reference to the object and the object has been deleted from the HSM, it is not possible to use that data archive again.
- Not all cipher suites in a standard Java configuration are available. They are limited to those listed in the `com.pingidentity.crypto.NcipherJCEManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Administrator's Manual

This manual provides information about using Ping Identity® 's PingFederate to deploy a secure Internet single sign-on (SSO) solution based on the latest security and e-business standards.

Key concepts

This section provides background information and preparation to help administrators understand and use PingFederate.

Tip: For an introduction to secure single sign-on (SSO), federated identity management, and Ping Federate product features, see *About identity federation and SSO* on page 26.

Connection types

PingFederate features an integrated administrative console for configuring four kinds of connections to identity-federation partners:

- Browser-based SSO – Also called Browser SSO in the administrative console, this term is often used to refer to standards-based secure SSO, which generally depends on a user's browser to transport identity assertions and other messaging between partner endpoints (see *Supported standards* on page 30).
- WS-Trust STS – Employs the PingFederate Security Token Service (STS), which enables Web service clients and providers (WSCs and WSPs) to extend SSO to identity-enabled web services at provider sites, using another set of standards (see the next section, *About WS-Trust STS* on page 100). 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) (see *About OAuth* on page 103).
- Provisioning – Provides automated cross-domain inbound and outbound user management (see *User provisioning* on page 125).

The types of connections can be configured together for the same partner or independently.

About WS-Trust STS

The PingFederate WS-Trust STS allows organizations to extend SSO identity management to web services. (For information about WS-Trust and the role of an STS, see *Web services standards* on page 47.)

The WS-Trust STS can be configured for partner connections independently or in conjunction with browser-based SSO for either an IdP or an SP deployment. The STS is bundled with separate plug-ins for standard SAML (Security Assertion Markup Language) token processing and generation (see *Token processors and generators* on page 100).

Connection-based policy

For both the IdP and SP roles, PingFederate employs a partner-connection configuration, which enables the association of web services authentication policies with federation partners. For STS processing, these policies define configurations for handling WS-Trust requests and transferring identity information between security domains (see *Web services standards* on page 47).

IdP configuration

In an IdP role, you use the administrative console to configure WS-Trust request-processing policy for your SP partner including:

- The type of SAML token to create—suitable for consumption by the intended web service provider (WSP, at the SP site)—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

In an SP role, you use the administrative console 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 that plug into the PingFederate server. These plug-ins deploy similarly to browser-based SSO adapters (see *SSO integration kits and adapters* on page 112).

For an IdP, token processors provide a mechanism through which PingFederate can validate an incoming token from a WSC and map attributes to be included in the issued SAML token.

For an 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.

Only SAML 1.1 or 2.0 tokens are generated by PingFederate configured as an IdP for sending across trust boundaries to a federated SP partner. Likewise, only SAML tokens are accepted by PingFederate configured as an SP. Token plug-ins allow a modular approach for validating and producing the various 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 STS token exchange within the same domain or trust boundary, you can also use the PingFederate STS to exchange one token type for another directly, without generating a transitional SAML token (see *Token translator mappings* on page 705).

PingFederate also allows you to use a configuration of a token processor or generator as a parent instance from which you can create child instances (see *Hierarchical plug-in configurations* on page 119).

Bundled token plug-ins

PingFederate is installed with token processors for an IdP configuration that accept and validate SAML 1.1, 2.0 tokens, OAuth Bearer access tokens, JWT tokens, Username tokens, and Kerberos tokens (see *Token models and management* on page 104). (SAML tokens are issued on the IdP side via built-in browser-based SSO capabilities.)

For an SP configuration, token generators are provided for issuing local SAML 1.1 or 2.0 tokens. (Incoming SAML tokens are validated, once again, by using built-in capabilities.)

Commercial token plug-ins

Ping Identity provides token plug-ins to work with various authentication systems and leading identity management systems. Available plug-ins, together known as *Token Translators*, may be downloaded 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 (WS clients or providers) to interact with the PingFederate STS.

In addition, for 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 at 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 (see *Attribute contracts* on page 121).

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 may be necessary, the PingFederate STS supports the use of customBinding.
>
> For more information about bindings, see Microsoft's *System-Provided Bindings* (docs.microsoft.com/en-us/dotnet/framework/wcf/system-provided-bindings).

Developers can obtain metadata from PingFederate to expedite configuring their applications. PingFederate offers two varieties of metadata, which are often used together to arrive at functional WSC and WSP configurations:

- STS Metadata Exchange at *`/pf/sts_mex.ping`*, which contains connection details relating to the SP partner.
- Federation Metadata at *`/pf/federation_metadata.ping`*, which contains details on the PingFederate public signing certificate and other information required to establish the trust relationship.

For more information about claim-based federated identity, see Microsoft's *A Guide to Claims–based Identity and Access Control* (msdn.microsoft.com/library/ff423674.aspx).

STS OAuth integration

PingFederate STS provides several ways to facilitate the use of issued tokens with an OAuth 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 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 (tools.ietf.org/html/rfc7523).

OAuth Access Token via 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 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 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 (tools.ietf.org/html/draft-ietf-oauth-saml2-bearer).

OAuth Access Token via 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 may have with an OAuth AS, allowing the client to obtain additional resources on behalf of already-authenticated users in follow-on transactions.

About OAuth

PingFederate can be configured to act as an OAuth authorization server (AS), allowing a resource owner (typically, an end user) to grant authorization to an OAuth client requesting access to resources hosted by a resource server (RS). The OAuth AS issues tokens to clients on behalf of a resource owner 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, please contact sales@pingidentity.com.

The PingFederate OAuth AS can be configured independently or in conjunction with STS or browser-based SSO for either an IdP or an SP deployment.

In an IdP deployment, an IdP adapter can be used to authenticate and provide user information for the access token.

In an SP deployment, the inbound SAML assertion can be used to provide authentication information about the user that can be associated with the access token through an OAuth attribute mapping in the IdP connection.

For an STS IdP, an OAuth token processor is provided with the PingFederate installation to validate incoming OAuth Bearer access tokens.

Delegated access types

Explicit delegation

This is the most common OAuth use case, which involves a resource owner (RO) who explicitly delegates to a client the authority 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 AS to issue access tokens for use in authenticating an API call. These tokens may be characterized by both their security model and data model.

Token security model

A token security model refers to the conditions that must be met by a client in order 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 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 (JavaScript Object Notation), 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 may require different token data models, token lifetimes, attribute contracts, token validation rules, or any combination of them, for various clients.

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 gives consent to the AS. The client subsequently exchanges the authorization code for an access token (and often a refresh token). Resource owner credentials are never exposed to the client.

Resource owner password credentials

```
password
```

The client collects the resource owner'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 is often returned with an access token. Once the original access token expires, the corresponding refresh token can be sent to the AS to obtain a fresh access token without requiring the resource owner to reauthenticate. 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 can only be used in conjunction with either the Authorization Code or Resource Owner Password Credentials grant type.

Implicit

```
implicit
```

An access token is returned to the client through a browser redirect in response to the resource owner authorization request (rather than an intermediate authorization code). This grant type is suitable 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 in order to obtain an access token. This access token is either associated with the client's own resources, and not a particular resource owner, or is associated with a resource owner for whom the client is otherwise authorized to act.

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 STS processing.

Client-initiated backchannel authentication (CIBA)

`urn:openid:params:grant-type:ciba`

The client presents an identity hint (and optionally a user code) to the AS. The AS identifies the resource owner based on the hint provided and then authenticates and obtains authorization from the resource owner via an out-of-band flow. Depending on the setup, the client can either poll 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 resource owner approves the authorization request, the AS returns a 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 in order to identify the device-authorization session and obtain an access token. This access token is associated with a resource owner for whom the client is otherwise authorized to act.

Validation grant type

`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 (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

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 users (the resource owner) approves, the authorization server issues an access token with such scopes.

Scopes are configured globally using the **OAuth Server**# **Scope Management** configuration wizard. Once defined, the availability of scopes can be managed 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. A static scope is defined by using a text value; for example, `read_bank_account`. A dynamic scope is defined by using a text value with a variable component represented by a wildcard; for example, `read_bank_account_txn:*`. As illustrated, 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.

Consent approval

With Authorization Code, Implicit, and Device Authorization grant types, an authorization server (AS) prompts the user (the resource owner) to grant authorization to share the user's information hosted by a resource server (RS). When granted, the AS issues an access token to the client. The client can then use the access token to access such information from the RS.

Default consent user interface

PingFederate handles the consent approval process by presenting the **Request for Approval** page to the resource owner by default. This page 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 is becoming more important to provide detailed information about the requests. While the scope description may 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 account (or accounts) to be made available to the client.

Client management and storage

OAuth clients interacts with an authorization server to obtain access tokens (and sometimes refresh tokens) for the purpose of accessing 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 (tools.ietf.org/html/rfc7591).

Client records are stored in XML files by default. This configuration provides administrators the capability to manage clients using the administrative console and the administrative API and developers to submit client creation requests based on the Dynamic Client Registration protocol specification. Client records are part of the configuration archive.

Alternatively, administrators can configure PingFederate to store client records externally, on a database server, a directory server, or some other storage medium through the use of the PingFederate SDK. (The OAuth Client Management Service requires client records to be stored externally.) Note that client records are not part of the 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 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.
- **None** when authentication is not required.

When deployed as an OpenID Connect Relying Party (RP), PingFederate can authenticate via client secret and private key JWT. It can also handle the scenario where authentication is not required.

Dynamic client registration

PingFederate supports dynamic client registration based on the OAuth 2.0 Dynamic Client Registration Protocol specification (tools.ietf.org/html/rfc7591). When enabled, it allows developers to register OAuth clients via an API based on open standards.

Transient grants and persistent grants

There are two types of OAuth authorization grants, namely transient grants and persistent grants.

Transient grants

Transient grants are valid only for the lifetime of their respective access tokens. Authorizations obtained by OAuth clients in the following manners are considered transient.

- Grants obtained by using **Authorization Code**, **Resource Owner Credentials**, or **Device Authorization** grant type, without the **Refresh Token** grant type.
- Grants obtained by using **Client Credential**, **JWT Bearer**, and **SAML 2.0 Bearer Assertion** grant type.

Transient grants are not preserved.

Persistent grants

Persistent grants typically bear a longer lifetime than their respective access tokens do. Authorization grants obtained by OAuth clients in the following manners are considered persistent.

- Grants obtained or updated by using the **Authorization Code**, **Resource Owner Credentials**, or **Device Authorization** grant type, in conjunction with the **Refresh Token** grant type.

  If the use cases involve mapping attributes from authentication sources (IdP adapter instances or IdP connections) or Password Credential Validator (PCV) instances to the access tokens (directly or through persistent grant extended attributes), such attributes and their values are stored along with the persistent grants so that they can be reused 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.

  If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens (directly or through persistent grant extended attributes), attribute values are obtained at runtime for each token request. No attributes or their values are stored with the persistent grants.

Persistent grant lifetime and maintenance

Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked or cleaned up.

Grants can persist without any expiration information. Grants can also persist with an idle timeout window, a maximum lifetime, or both. If an idle timeout value is configured, the idle timeout window slides when a persistent grant is updated. When an idle timeout value is configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity, or are revoked or removed. When an idle timeout value is configured with a maximum lifetime, persistent grants remain valid until they expire (due to inactivity or lifetime expiration) or are removed from the grant storage.

PingFederate removes expired grants and the associated attributes from the grant data store once a day. The frequency and the size of the cleanup batch are configurable. Optionally, PingFederate can put a cap on 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.

ⓘ **Important:** PingFederate uses a pre-installed HSQLDB database as its grant data store after the initial setup. We strongly recommend to use a secured external storage for production deployments (see *Defining an OAuth grant data store* on page 186).

Grant storage and management

PingFederate uses a pre-installed HSQLDB database as its persistent grant data store after the initial setup. We strongly recommend to use a secured external storage for production deployments. Supported storage platforms database server, directory servers, and other storage solutions through the use of the PingFederate SDK.

Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked or cleaned up.

PingFederate removes expired grants and the associated attributes from the grant data store once a day. The frequency and the size of the cleanup batch are configurable. Optionally, PingFederate can put a cap on 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 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.

Grant-management endpoint

The grant-management endpoint allows resource owners to view and optionally revoke the persistent access grants they have authorized.

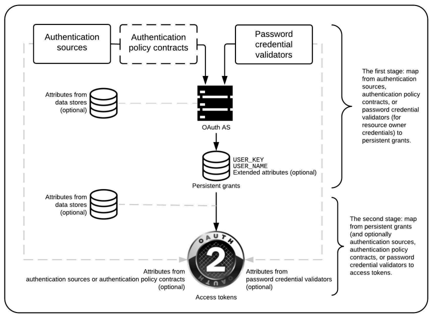
The token revocation endpoint is intended for OAuth clients; this is the endpoint, to which clients send their token revocation requests. The grant-management endpoint is for resource owners. It displays a list of grants the resource owners have made. Resource owners can view and optionally revoke one or more grants as they see fit.

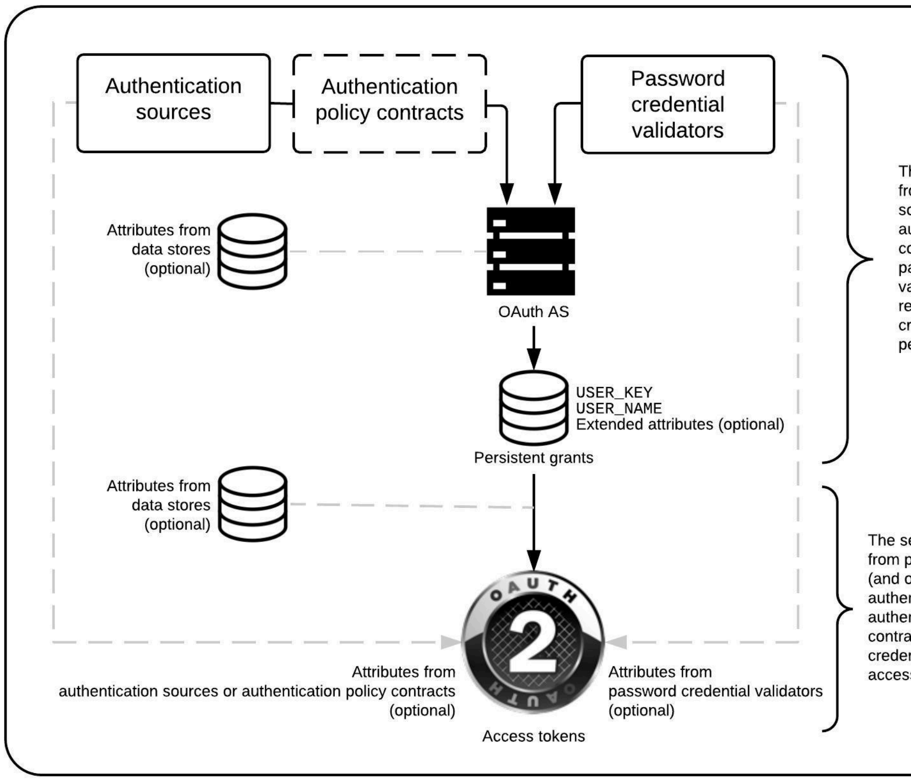
Mapping OAuth attributes

Mapping OAuth attributes is a two-stage processing workflows:

- The first stage: map from authentication sources (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 (and optionally authentication sources, authentication policy contracts, authentication context, or Password Credential Validator instances to access tokens.

Note that 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.


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

To accomplish the first stage, mapping is required for setting up persistent grants, including a user key and all extended attributes.

The mappings may use attributes obtained during initial authentication events within PingFederate, namely attributes from IdP adapter instances, attributes from assertions via IdP connections, attributes from authentication policy contracts, or attributes returned by password credential validator instances. Moreover, data store queries may also be configured; for example, to retrieve 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, if you have two Active Directory domains, the sAMAccountName attribute value of an end user in one domain may collide with that of another end user in the other domain. In this scenario, 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.

You may also set up specific mappings between the authentication sources, authentication policy contracts, or password credential validators. When the authentication context matches a specific mapping, attributes from the authentication sources, authentication policy contracts, or password credential validators can be mapped into the access tokens. Additionally, you can use an expression to retrieve from the **HTTP Request** Java object the authentication method that a client uses (or the private key JWT with which a client authenticates if the client uses the private_key_jwt authentication method) and then map it into the access tokens.

Data stores may also be used here to retrieve any required user attributes.

Runtime processing

At runtime, the first time a client requests an OAuth token the two mapping sequences are employed sequentially. The second mapping is invoked every time a new access token is requested based on an existing persistent grant.

OAuth user-facing screens

The PingFederate OAuth AS provides five screens presented to end users (the resource owners) during OAuth transactions, a page prompting for approval of the requested scopes, another page providing a means of revoking persistent access grants, and finally three other pages displaying information for the purpose of connecting OAuth-capable IOTs based on the *OAuth 2.0 Device Authorization Grant* (tools.ietf.org/html/draft-ietf-oauth-device-flow). As needed, administrators can customize these screens.

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 (see *openid.net/connect*).

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 is the release of at least two tokens to the requesting client application: an ID token and an OAuth access token. (Depending on associated grant types, a refresh token may also be released.)

The ID token is an integrity-secured, self-contained token in JSON Web Token (JWT) format containing claims about the user, namely the subject. A client uses the ID token to securely identify the user authenticated by an OP accessing the client application. 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 `/pf-ws/rest/sessionMgmt/revokedSris`).

As an OP, PingFederate can optionally accepts request parameters via self-contained, signed JWTs. This capability enables PingFederate to validate the integrity of the request parameters it receives before processing the request further. Furthermore, it is also capable of including a state hash (s_hash) in the ID token to protect the integrity of the state parameter.

Relying Party support

As an 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 client application, an OAuth client.

The setup involves establishing an IdP connection to the OP. In essence, PingFederate retrieves identity information from the OP and passes the end-user claims, which are basically 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, a modern standard that has been gaining momentum in the industry.

PingFederate is also capable of sending request parameters via 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.

CORS support for OAuth endpoints

PingFederate supports cross-origin resource sharing (CORS) 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`

As needed, administrators can add or remove allowed origins using the administrative console on the **Authentication Application** page. For more information, see *Configure an authentication application* on page 492. 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 JSON Web Key Sets.

SSO integration kits and adapters

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 SSO.

Note: See the PingFederate *SSO Integration Overview* on page 924 for more information.

For an IdP (the first mile), this integration process involves providing a mechanism through which PingFederate can look up a user's current authenticated session data (for example, a cookie) or authenticate a user without such a session. For an 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 plug-in *authentication selectors*, which enable dynamic selection of authentication sources based on administrator-specified criteria.

PingFederate also includes a robust software development kit (SDK), which software developers can use to write their own adapters, data stores, and other components, for specific systems.

Bundled adapters

PingFederate comes bundled with a set of adapters.

Identifier First Adapter

When a variety of user types are authenticating 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. See *Identifier First Adapter* on page 710.

HTML Form Adapter

Used in conjunction with Password Credential Validators. These adapters provide integration with user-data stores in directory servers or locally. See *HTML Form Adapter* on page 716.

Kerberos Adapter

Provides a seamless desktop SSO experience for Windows environments and supports authentication mechanism assurance from Active Directory domain service. This adapter is recommended for new configurations as a simpler alternative to the separately available IWA Integration Kit. See *Kerberos Adapter* on page 730.

OpenToken Adapter

Provides a generic interface for integrating with various applications, including Java- and .NET-based applications. See *OpenToken Adapter* on page 734.

Composite Adapter

Allows multiple configured IdP adapters to execute in sequence. This capability, called *adapter chaining*, may be used either for single-adapter usage, depending on authentication context, or to support multifactor authentication via a series of adapters. See *Composite Adapter* on page 740.

HTTP Basic Adapter

Used in conjunction with Password Credential Validators. These adapters provide integration with user-data stores in directory servers or locally. See *HTTP Basic Adapter* on page 743.

PingID®

PingID is a cloud-based authentication service that binds user identities to their devices, making it an effective multifactor authentication solution. See *PingID documentation*.

Bundled authentication selectors

PingFederate provides plug-in 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: The results of authentication-selection criteria evaluation can be used to select subsequent selectors or authentication sources, which allows handling of complex hierarchical access-policy decisions (see *Authentication policies* on page 324).

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 a specified range, or 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. See *Configuring the CIDR Authentication Selector* on page 326.

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, this selector allows you to choose whether or not Integrated Windows Authentication is attempted based on the PingFederate cluster node with which a Key Distribution Center is associated. See *Configuring the Cluster Node Authentication Selector* on page 327.

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. See *Configuring the Connection Set Authentication Selector* on page 327.

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. See *Configuring the Extended Property Authentication Selector* on page 328.

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. See *Configuring the HTTP Header Authentication Selector* on page 330.

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. See *Configuring the HTTP Request Parameter Authentication Selector* on page 331.

OAuth Client Set Authentication Selector

The 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. See *Configuring the OAuth Client Set Authentication Selector* on page 333.

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. See *Configuring the OAuth Scope Authentication Selector* on page 334.

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. See *Configuring the Requested AuthN Context Authentication Selector* on page 335.

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. See *Configuring the Session Authentication Selector* on page 336.

ⓘ **Note:** Authentication selectors rely on HTTP requests, HTTP headers, POST data, or a combination of them. Ensure that standard security measures are in place when using these selectors.

Integration kits

Ping Identity regularly develops and maintains integration kits, including adapters, to work with applications and leading identity management systems. Available kits may be downloaded from the Ping Identity Downloads website. Additional authentication selectors may also be added to the download site periodically; contact sales@pingidentity.com if you are looking for specific authentication-selection capabilities.

Software development kit (SDK)

The PingFederate SDK provides a flexible means of creating custom adapters to integrate federated identity management into your system environment. See the PingFederate *SDK Developer's Guide* on page 931.

Security infrastructure

This section describes the PingFederate security infrastructure that supports encrypted messaging, certificates, and digital signing. These functions are integrated into PingFederate's configuration screens 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. Digital signatures are used in both SAML (including STS tokens) and WS-Federation electronic documents.

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 owner of the certificate along with a public key. Applying a digital signature creates and encrypts a hash from the message you are signing, using your private key. PingFederate provides a choice of signature encryption algorithms when a stronger algorithm is required.

To ensure the integrity of SAML messages or 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; XML encryption is used for this purpose.

The certificate should be signed by a Certificate Authority (recommended), but it can be self-signed or signed by an untrusted third party. After generating a keypair and a self-signed certificate, you can use PingFederate to create a Certificate Signing Request (CSR) and send it to a CA for signing. After the CA has generated a Certificate Signing Response, you can import it into PingFederate's certificate management system. (The CA's certificate must be in PingFederate's trusted store or in the Java runtime `cacerts` store.)

PingFederate enables signing and validation of requests and responses. In addition, PingFederate provides for certificate generation, import and export functionality, CSR generation, and application of digital signatures. You can create reusable global signing certificates across your federated connection base and import signature verification certificates for each partner (see *Manage digital signing certificates and decryption keys* on page 299).

ⓘ **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, both when they are initially imported and at runtime when they are used to encrypt, decrypt, and digitally sign or verify assertions.

PingFederate can also check 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 SSO or 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 is acting as the client making an HTTPS request to a separate server

If a certificate is expired or revoked, the associated SSO or SLO transaction fails at runtime and an error is written to the transaction log. In the administrative console, an expired or revoked certificate is identified as such in the Status column of its respective Certificate Management list.

CRL revocation checking

This process involves querying a CRL distribution-point URL and ensuring that a certificate is not on the returned revocation list maintained at the site. The URL is specified in the certificate.

No setup is needed in the administrative console to enable CRL checking. PingFederate automatically checks CRLs if all of the following conditions are met:

- The certificate contains the URL where the CA maintains its CRL.
- The URL is accessible.
- The returned CRL is signed and the signature verified.
- CRL validation is not explicitly disabled as a failover option in the OCSP setup.

OCSP revocation checking

OCSP was developed as an alternative to CRL validation and provides a more centralized and potentially more reliable means of checking certificate status. In this scenario, an OCSP Responder URL is normally embedded in the incoming certificate (a configured default URL may be used, alternatively). The URL, maintained by the issuing CA, is used 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 revocation checking is handled by the Responder server, which returns the certificate status.

A PingFederate administrator can enable and configure OCSP processing in the administrative console. The protocol may be used exclusively or in conjunction with CRL checking as a backup.

For more information about OCSP, see *tools.ietf.org/html/rfc2560*.

For configuration steps, see *Configure certificate revocation* on page 311.

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, the protocol specifications require signatures; for example, all SAML STS tokens and all SSO assertions sent across the POST binding must be signed. (These requirements are enforced by the PingFederate administrative console and the runtime protocol engine.) Other uses of the digital signatures are optional between partners and enforced if specified for a partner connection.

If a digital-signing certificate is not issued by a trusted CA (that is, "self-signed"), then the signing partner must send the public-key certificate out-of-band (for example, via email) to the partner. The partner must import the certificate into PingFederate when configuring a connection to the signing partner for SSO/SLO or STS.

If the certificate is signed by a trusted CA and the signing partner chooses to embed the certificate in all signed messages, then the verifying partner can elect to use the embedded certificate for signature verification, after validating it against the Subject DN of the original certificate. The public-key certificate may or may not be sent out-of-band (just the Subject DN is required).

Tip: PingFederate can extract the Subject DN from the certificate, when available, during the signature-verification configuration.

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. Note that for each trust model, PingFederate always verifies that the certificate is current and that the signature in the message can be verified using 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 the anchored trust model is chosen, 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

When this option is chosen, incoming signatures are verified exclusively using the certificates imported for a connection into PingFederate (or a secondary, backup certificate when specified). The certificate may be self-signed or issued by a trusted CA. The certificate chain, if any, is not verified. However, revocation checking, when enabled, is performed up any existing chain as far as available.

Secure sockets layer

SSL certificates signed by a CA can be used to identify one or both ends of the federation. SSL/TLS provides an encrypted connection between the two parties in which the content of a message is not exposed, thus ensuring 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 IdP site. On the SP side, the Artifact Resolution Service should also use SSL/TLS. Optionally, SSL/TLS may also be used to secure communication between internal data stores and PingFederate and between the PingFederate STS and web service client or provider applications.

The SSL/TLS server-client handshake involves negotiating cipher suites to be used for encryption and decryption on each side of a secured transaction. Cipher suites are stored 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`

These cipher-suite configuration files are located in the `<pf_install>/server/default/data/config-store` directory. Weaker cipher suites are commented out in these files. Retain this cipher-suite configuration to ensure the most secure transactions.

Important: Due to the import restrictions of some countries, Oracle Server JRE (Java SE Runtime Environment) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see the Java 8 release notes from Oracle (www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html).

For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide (docs.oracle.com/en/java/javase/11/migrate/).

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 may enable the same selection mechanism as well (see *Managing cipher suites* on page 277).

Authentication

Three methods of authentication, described below, are available for use with PingFederate for browser-based SSO to authenticate connection partners making SOAP requests. For SOAP authentication by STS clients, a separate option using either or both of the first two methods, may be configured (the third method, digital signing, is automatically required). The selection of one or more method(s) must be agreed upon between partners and synchronized 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's certificate out-of-band (see *Manage SSL client keys and certificates* on page 297).

Digital signatures

Partners sign the XML message transmitted via the SSL/TLS connection. Signatures are verified by the receiver based upon the certificate(s) configured for that connection. Each partner should import the other's certificate(s) out-of-band (see *Manage digital signing certificates and decryption keys* on page 299).

Trusted certificates

PingFederate validates the trust of all certificates. A certificate is trusted if the certificate of its issuer is in PingFederate's trusted certificate store. The root certificate of the CA, by which a certificate is issued, must be imported into PingFederate's trusted certificate store or contained in the Java runtime `cacerts` store.

Encryption

PingFederate supports the optional SAML 2.0 specification allowing for encryption of assertions (including STS SAML tokens), which further enhances confidentiality when required.

For SAML 2.0 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 plug-in configurations

PingFederate allows you to use a configuration of an adapter (as well as certain other PingFederate plug-ins) as a *parent* instance from which you can create *child* instances. You can then modify the inherited configuration for the child instances as needed. This feature provides easier management of adapter settings in cases where only small changes to an existing adapter (or plug-in) configuration need to be made for a particular use case.

For example, different SP-connection adapter instances might have their own IdP logon 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: You can also override adapter instances as part of mapping them into either SP or IdP connections, for cases where overridden settings may apply only to one particular connection configuration.

Any changes to a parent configuration are propagated to its child (or connection-based) configurations provided the changes are not already overridden in the derived instance.

In addition to adapters, PingFederate allows you to create parent/child configurations for the following plug-in types:

- Token Translators (see *Token processors and generators* on page 100)
- Access Token Management instances (see *Access token management* on page 445)
- Password Credential Validators (see *Managing Password Credential Validator instances* on page 200)
- Identity Store Provisioners (see *Configure Identity Store Provisioners* on page 587)

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 (the assertion) containing user-identity information that a service provider (SP) can translate, or map, to local user stores.

For browser-based SSO, PingFederate enables two modes of identity mapping between domains: account linking and account mapping.

For WS-Trust STS, account mapping is used.

Refer to subsequent topics for more information about these identity mapping options.

Account linking

Under the standards, *account linking* can be used for browser-based 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*, may be either a unique attribute, such as an email address, or a *pseudonym* generated by the IdP to uniquely identify individual users. Pseudonyms can be used when privacy is a concern; they cannot easily be traced back to a user's identity at the partner site.

During the user's first SSO request, the 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.

Tip: PingFederate in the SP role uses a default, HSQLDB database to handle account linking. You can use your own data store instead, as needed. For more information, see *Defining an account-linking data store* on page 197.

Optionally, additional attributes may be sent with the name identifier. When a pseudonym is used as the account link, however, care must be taken to send only general attributes (a user's organizational role or department, for example) 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* (see */sp/defederate.ping* on page 764). A user who has defederated may later elect to re-associate with a local user account.

SP affiliations

Under the SAML 2.0 specifications, an IdP can configure PingFederate to enable a group of SPs—an *SP affiliation*—to share the same persistent name identifier (see *SP affiliations* on page 556). This capability facilitates the use case in which a number of business partners have an existing relationship and sharing a single name identifier among all parties reduces the federation integration effort.

Account mapping

Account mapping (also called "*attribute mapping*") enables an 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. The attributes used to look up the user are always "exposed"; that is, they are known to both the IdP and SP. An email address, for example, is a commonly used identifying attribute.

Account mapping can be used to achieve one-to-one mapping (individual user accounts exist on both sides of federated connection) or many-to-few (IdP users without accounts at destination sites may be mapped to guest accounts or to a role-based general account).

For browser-based 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, whereby the user is mapped 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 may be sent with the transient identifier. Again, care should be taken to preserve privacy.

Account mapping is commonly implemented in B-to-B or B-to-E use cases where it might be appropriate for the administrator to create a user lookup on behalf of the user.

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 IdP can pass other user attributes in an assertion (including SAML tokens for web services). This supplemental information can be used by the SP for several purposes. For example, attributes may be used to map and authorize the user into a specific role, with associated site permissions. In other cases, attributes may be used to customize the end application display for a more robust user experience.

The SP also has the option of incorporating additional attributes prior to creating a session for the target application. This is commonly done where the SP also maintains an account for the user and wants to pass 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 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 tokens (JWTs), 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 (see *Attribute masking* on page 123).

For an IdP or an OpenID Provider (OP), the attribute contract defines which attributes PingFederate sends in an assertion, a JWT, or an ID token. While this contract is fixed for all users authenticating to the partner, the values used to fulfill the contract may differ from one user to the next. The attribute contract may be fulfilled by relying on a combination of different data sources:

- The IdP adapter or STS token processor
- An IdP attribute source, which identifies the location of individual attributes in a data store
- Static text values for some attributes, or text values combined with variables
- Expressions (see *Attribute mapping expressions* on page 859)

For an 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. PingFederate can be configured to pass these attributes to the SP adapter or, for web services, to the SP token generator (see *Manage SP adapters* on page 583 or *Manage token generators* on page 691). You can also use attributes to look up additional attributes in local data stores, which may be needed to start a user session or create a local security token for web services (see *Adapter contracts* on page 122 or *STS token contracts* on page 123).

The attribute contract always contains the user identifier (SAML_SUBJECT in a SAML assertion and sub in a JWT or an ID token)—the primary information used to identify the user—unless you are using account linking for browser-based 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, a single attribute contract can be created for the IdP connection partner. This single contract would be constructed to supply all the attributes needed by both SP adapters.

Name formats

By agreement with an SP partner, an IdP may specify a format (email, for example) associated with the SAML_SUBJECT. The SP may require this information to facilitate handling of the format.

The partner agreement may also include a requirement for the IdP to provide format specifications associated with other attributes.

For browser-based SSO connections, 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 can also define a customized selection of additional attribute formats (see *Set up an attribute contract* on page 510).

ⓘ **Note:** The designation of formats is not applicable to SP administrators. The information is simply available in the incoming assertion to an SP application that might need it for particular processing requirements.

For the WS-Trust IdP configuration, attribute-name formats cannot be specified. If needed, however, an administrator can use a special variable in the attribute contract to set the subject-name format (see *Define an attribute contract for IdP STS* on page 682). (The same variable is also available for browser-based SSO attribute contracts, but the feature is deprecated.)

STS namespaces

By agreement with an SP partner for a WS-Trust STS connection, an IdP may specify an XML namespace to be associated with an attribute (for example, to use claims-based authorization with WIF clients—see *WSC and WSP support* on page 101). Namespaces can be specified only for attributes of a WS-Trust IdP configuration using **SAML 1.1** or **SAML 1.1 for Office 365** as the default token type (see *Define an attribute contract for IdP STS* on page 682).

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 IdP side of a federation, adapter attributes are supplied to PingFederate by an IdP adapter (see *SSO integration kits and adapters* on page 112 and *Manage IdP adapters* on page 484).

On the SP side, adapter contract attributes are those required by an adapter to start a session with an application. At least one *adapter type* is needed for each security domain. Then an *adapter instance* must be configured for each target application. (See *Manage SP adapters* on page 583.)

Adapter contracts on the SP side are fulfilled using attributes from the attribute contract, possibly enhanced through other attributes looked up from local data stores. For example, if several target applications are controlled by the same security context and can receive 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 (see *Manage target session mappings* on page 605).

Extended adapter contract

Adapter contracts are created when an adapter type is deployed with PingFederate. When developed, these adapters are “hard-wired” to look up or set a specific set of attributes. After deployment, your attribute requirements may change. To streamline adjustment of adapter contracts, PingFederate allows an administrator to add additional attributes to the adapter instance through the administrative console. These adjustments are called *extended adapter contracts*.

STS token contracts

Similar to an adapter contract for browser-based SSO, an 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 IdP side of a federation, token-processor attributes are supplied to PingFederate (see*Token processors and generators* on page 100 and *Manage token processors* on page 675).

On the SP side, token-generator contract attributes are those required by a token generator to pass identity information from the token to the web service client application. At least one token generator type is needed for each security domain. Then a token generator instance must be configured for each target application (see *Manage token generators* on page 691). 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 (see *Manage SP token generator mappings* on page 694).

Extended token generator contract

Token-generator contracts are created when a token-generator type is deployed with PingFederate. When developed, these token generators are “hard-wired” to look up or set a specific set of attributes. After deployment, your attribute requirements may 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*.

Data stores

Data stores represent external systems where user attributes and other data are stored. Once defined, you can configure PingFederate to retrieve user attributes from data stores for contract fulfillment and token authorization in various use cases. You can also configure PingFederate to write certain records or log messages to data stores. PingFederate supports a wide variety of database servers and directory servers. As needed, you or developers at your organization can create custom drivers through the PingFederate SDK for connecting to other kind of data repositories, such as flat files or SOAP-connected databases. For more information, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.

Attribute masking

At runtime PingFederate logs user attributes (see *PingFederate log files* on page 224). To preserve user privacy, you may wish to mask the values of logged attributes.

PingFederate provides this masking capability at all points where the server logs attributes. These points include:

- Data-store lookup at either the IdP or SP site (see *Managing data stores* on page 166).
- Retrieval of attributes from an IdP adapter or token processor (see *Set pseudonym and masking options* on page 486 and *Set attribute masking* on page 679).

- SP-server processing of incoming attributes based on the SSO attribute contract, (see *Define an attribute contract* on page 604).

  Note that 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 (see *Account linking* on page 120).
- SP-server processing of incoming attributes in response to an Attribute Request under XASP (see *Configure security policy for Attribute Query* on page 626).

  For information about XASP, see *Attribute Query and XASP* on page 45.

  Important: Many adapter implementations, as well as other product extensions, may independently write unmasked attribute values to the PingFederate server log. These implementations are beyond the control of PingFederate. If sensitive attribute values are a concern when using such a component, a system administrator can adjust the component's logging threshold in `log4j2.xml` to prevent the recording of attributes.

About token authorization

PingFederate provides an optional configuration to evaluate user attributes, as well as other runtime variables (such as authentication context), for authorization purposes. This feature, known as *token authorization*, provides a way for administrators to extend access policy directly to many areas, such as Browser SSO, STS, and OAuth events, by conditionally allowing or disallowing the issuance of relevant security tokens (for example, SAML assertions, STS tokens, OAuth access tokens, or session cookies). The option is also available for extending authorization policy to attribute-query responses, IdP adapter contracts, and authentication policy contracts.

Administrators can configure token authorization using Issuance Criteria screens immediately following the configuration of attribute mapping at all applicable points in the administrative console (see *Define issuance criteria for IdP Browser SSO* on page 519, as an example).

Issuance criteria

The token-authorization configuration consists of one or more rules that evaluate attribute values against selected conditions. You can choose from among several sources for the attributes, depending on the type of configuration that contains the token-authorization setup. The sources always consist of all of those available for attribute mapping, including data stores (when configured) and runtime information related to the context of an event. In addition, values of mapped attributes are available 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 is halted, a configurable **Error Result** code is passed back up through the system, 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.

User provisioning

PingFederate provides cross-domain user provisioning and account management. User provisioning is an important aspect of identity federation. Often 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 may be configured either in conjunction with partner SSO connections or separately:

- At an IdP site, you can automatically provision and maintain user accounts at service-provider sites that have implemented the System for Cross-domain Identity Management (SCIM), or at selected SaaS providers (see the next section, *Outbound provisioning for IdPs* on page 125).

  For information about SCIM, please refer to *www.simplecloud.info*.
- When PingFederate is configured as an SP, you can provision and manage 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 (see *Provisioning for SPs* on page 126).

Outbound provisioning for IdPs

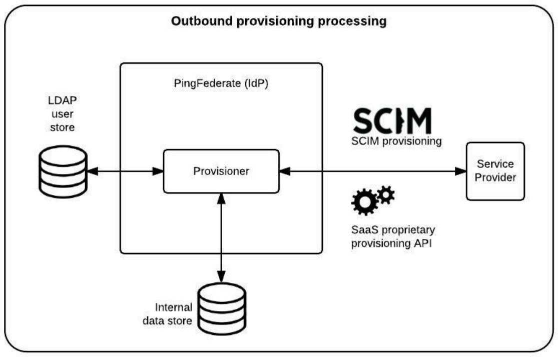
User provisioning is an important aspect of identity federation. Often when organizations enable SSO for their users, they must ensure that some form of account synchronization is in place. Automated user provisioning features within PingFederate free administrators from having to devise a manual strategy for this.

For IdP sites, PingFederate provides built-in automated provisioning and user-account management to SCIM-enabled services providers and to selected SaaS providers, via their proprietary provisioning APIs.

Outbound provisioning also provides an automated means of account disabling or deprovisioning, which may 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.

When outbound provisioning is 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, and Oracle Directory Server; templates are used to pre-configure many provisioning settings. Although these are the only data stores formally tested and supported, other LDAP data stores will likely work as well.

Internal data store

Tested internal data stores used for synchronization include HSQLDB, Microsoft SQL Server, Oracle Databases, Oracle MySQL, and PostgreSQL. A demonstration-only, embedded HSQLDB database is installed by default. Again, any relational database may be used; scripts are provided to aid setup.

Provisioning for SPs

User provisioning is an important aspect of identity federation. Often 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.

When configured as an SP, PingFederate offers two provisioning options:

- SCIM inbound provisioning for SCIM requests coming either from inside or outside an organization
- Just-in-time provisioning for creating and updating user accounts based on information contained in SSO tokens

Inbound provisioning

SCIM inbound provisioning provides support for incoming SCIM messages containing requests to create, read, update, or delete (or deactivate) user and group records in Microsoft Active Directory data stores or custom user stores via the Identity Store Provisioners. PingFederate supports SCIM attributes in the core schema and custom attributes through a schema extension. An administrator can configure this provisioning feature by itself or in conjunction with an SSO or other connection types.

In effect, inbound provisioning provides an organization with a dedicated SCIM service provider, which can route user-management requests to an organization's centralized user store. The requests may originate from trusted applications within an organization (for example, a human-resources on-boarding SaaS product) or from trusted partner IdPs.

For setup information, see *Configure SCIM inbound provisioning* on page 635. To integrate inbound provisioning with custom user stores, see *Configure Identity Store Provisioners* on page 587. For application-development information about using PingFederate endpoints for SCIM provisioning, see *SCIM inbound provisioning endpoints* on page 766.

Just-in-time provisioning

At an SP site, PingFederate can create and update local user accounts in an external LDAP directory or Microsoft SQL Server as part of SSO processing—*Just-in-time (JIT) provisioning* (also formerly known as Express Provisioning). This feature allows SPs to maintain accounts for users who authenticate via IdP partners without having to provision accounts manually, when local accounts are required.

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, assertion attributes can be supplemented with user attributes returned from an Attribute Query.

PingFederate can also update existing user accounts based on assertions. When this option is enabled, PingFederate can add or overwrite attributes for a local user account each time SSO for a user is processed.

Note: Note that once user attributes are provisioned, they cannot be removed using JIT provisioning. Where deprovisioning is required, we recommend using SCIM inbound provisioning.

For information about enabling JIT Provisioning, see *Choose IdP connection options* on page 597. For configuration information, see *Configure just-in-time provisioning* on page 627.

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 via 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.

Federation hub use cases

PingFederate can be configured 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 can bridge browser-based SSO between identity providers and service providers. It stands in the middle of the SSO and SLO flow, acting as the SP for the identity providers and as the 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.

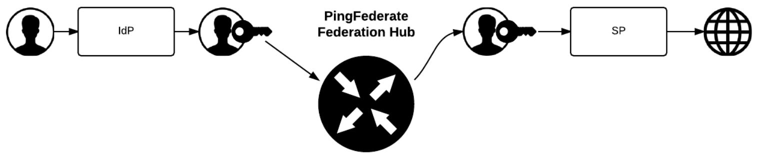
The federation hub capability can be deployed alongside with other OAuth use cases, IdP connections, SP connections, or any combination of them, to your partners. This flexibility helps in streamlining your federation infrastructure and reducing operating costs.

Bridging an IdP to an SP

About this task

In this use case, PingFederate is bridging SSO and SLO transactions between an identity provider and a service provider. For example, you may have a legacy IdP system that is only capable of sending SAML

1.1 assertions via POST. Your service provider however requires SAML 2.0 assertions via the artifact binding. With federation hub, you can configure PingFederate to consume inbound SAML 1.1 assertions (by POST), translate them to SAML 2.0 assertions, and send them via the artifact binding to the service provider.



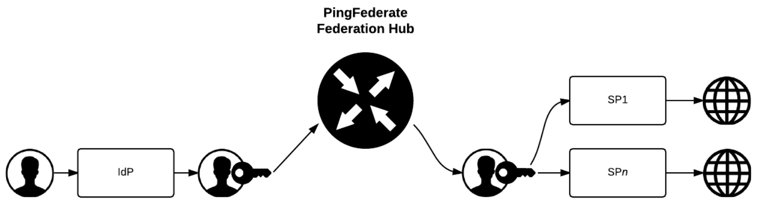
Steps

1. Enable both the IdP and the SP roles with the applicable protocols on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
2. Create a contract to bridge the attributes between the identity provider and the service provider (see *Federation hub and authentication policy contracts* on page 131).
3. Create an IdP connection between the identity provider and PingFederate (the federation hub as the SP) and add to the IdP connection the applicable authentication policy contract(s) on the **Target Session Mapping** screen.
4. Create an SP connection between PingFederate (the federation hub as the IdP) and the service provider and add to the SP connection the corresponding authentication policy contract on the **Authentication Source Mapping** screen.
5. Work with the identity provider to connect to PingFederate (the federation hub) as the SP.
6. Work with the service provider to connect to PingFederate (the federation hub) as the IdP.

Bridging an IdP to multiple SPs

About this task

In this use case, PingFederate is bridging SSO and SLO transactions between an identity provider and multiple service providers. For example, your company wants to route federation requests from a recently acquired subsidiary through its federation infrastructure. With PingFederate, you can multiplex one IdP connection to multiple SP connections to the desired service providers. The federation hub consumes assertions from the subsidiary and creates new assertions to the respective service providers.



Steps

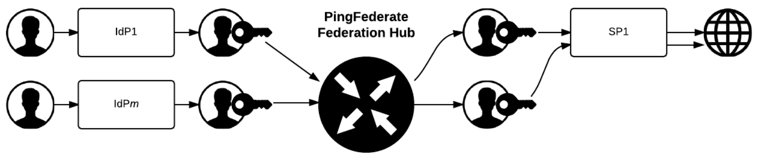
1. Enable both the IdP and the SP roles with the applicable protocols on the **System**# **Protocol Settings Settings**# **Roles & Protocols** screen.
2. For each service provider, create a contract to the identity provider (see *Federation hub and authentication policy contracts* on page 131). Multiple contracts are likely required, because each service provider may require a unique set of attributes.
3. Create an IdP connection between the identity provider and PingFederate (the federation hub as the SP) and add to the IdP connection the applicable authentication policy contract(s) on the **Target Session Mapping** screen.

4. For each service provider, create an SP connection between PingFederate (the federation hub as the IdP) and the service provider and add to the SP connection the corresponding authentication policy contract on the **Authentication Source Mapping** screen.
5. For each service provider supporting the SAML IdP-initiated SSO profile, map the expected target resources to the corresponding SP connections on the **Service Provider**# **Target URL Mapping** screen.
6. Work with the identity provider to connect to PingFederate (the federation hub as the SP).
7. Work with each service provider to connect to PingFederate (the federation hub as the IdP).

Bridging multiple IdPs to an SP

About this task

In this use case, PingFederate is bridging SSO and SLO transactions between multiple identity providers and a service provider. 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, translates SAML assertions from the business partners to WS-Federation security tokens and send them over to SharePoint.



Steps

1. Enable both the IdP and the SP roles with the applicable protocols on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
2. Create a contract to bridge the attributes between the identity providers and the service provider (see *Federation hub and authentication policy contracts* on page 131). You likely need only one contract unless the service provider requires a different set of attributes from each identity provider.
3. For each identity provider, create an IdP connection between the identity provider and PingFederate (the federation hub as the SP) and add to the IdP connection the applicable authentication policy contract(s) on the **Target Session Mapping** screen.
4. On the **Selectors** screen, configure an authentication selector (for example, an instance of the *Identifier First Adapter* on page 710) to map each identity provider to the corresponding IdP connection in an authentication policy.
5. Create an SP connection between PingFederate (the federation hub as the IdP) and the service provider and add to the SP connection the corresponding authentication policy contract on the **Authentication Source Mapping** screen.

Important: PingFederate includes the Entity ID of the original identity provider (Authenticating Authority) in SAML 2.0 assertions so that the service provider can determine the original issuer of the assertions. This is especially important when bridging multiple identity providers to one service provider

—the service provider 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** screen and then map **Context: Authenticating Authority** as the attribute value on the **Attribute Contract Fulfillment** screen.

For information about Authenticating Authority, see section *2.7.2.2 Element <AuthnContext>* in the SAML 2.0 specification (https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)

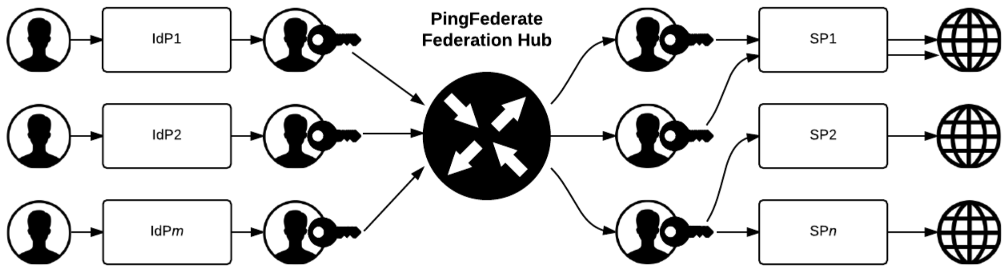
Note: If the service provider does not take action based on Authenticating Authority, depending on the attributes from the identity providers, you may define validation rules on the **Issuance Criteria** screen to protect against user impersonation between IdPs.

6. Work with each identity provider to connect to PingFederate (the federation hub as the SP).
7. Work with the service provider to connect to PingFederate (the federation hub as the IdP).

Bridging multiple IdPs to multiple SPs

About this task

This PingFederate federation hub use case is a combination of *Bridging an IdP to multiple SPs* on page 128 and *Bridging multiple IdPs to an SP* on page 129.



Steps

1. Enable both the IdP and the SP roles with the applicable protocols on the **System# Protocol Settings# Roles & Protocols** screen.
2. Create multiple contracts to bridge the attributes between the identity providers and the service providers (see *Federation hub and authentication policy contracts* on page 131).
3. For each identity provider, create an IdP connection between the identity provider and PingFederate (the federation hub as the SP) and add to the IdP connection the applicable authentication policy contract(s) on the **Target Session Mapping** screen.
4. On the **Selectors** screen, configure an authentication selector (for example, an instance of the *Identifier First Adapter* on page 710) to map each identity provider to the corresponding IdP connection in an authentication policy.
5. For each service provider, create an SP connection between PingFederate (the federation hub as the IdP) and the service provider and add to the SP connection the corresponding authentication policy contract on the **Authentication Source Mapping** screen.

Important: PingFederate includes the Entity ID of the original identity provider (Authenticating Authority) in SAML 2.0 assertions so that the service provider can determine the original issuer of the assertions. This is especially important when bridging multiple identity providers to one service provider

—the service provider 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** screen and then map **Context: Authenticating Authority** as the attribute value on the **Attribute Contract Fulfillment** screen.

For information about Authenticating Authority, see section *2.7.2.2 Element <AuthnContext>* in the SAML 2.0 specification (https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)

Note: If the service provider does not take action based on Authenticating Authority, depending on the attributes from the identity providers, you may define validation rules on the **Issuance Criteria** screen to protect against user impersonation between IdPs.

6. For each service provider supporting the SAML IdP-initiated SSO profile, map the expected target resources to the corresponding SP connections on the **Service Provider**# **Target URL Mapping** screen.
7. Work with each identity provider to connect to the federation hub (as the SP).
8. Work with each service provider to connect to the federation hub (as the IdP).

Federation hub and authentication policy contracts

PingFederate uses two connections to bridge an identity provider to a service provider:

- 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

It fuses these two connections together by using an authentication policy contract (formerly known as connection mapping contract) as the medium to carry user attributes from the identity provider to the service provider.

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 have the option to configure the IdP connections, SP connections, or both, to look up values from multiple data store instances.

When bridging one identity provider to one service provider, you need to create one authentication policy contract and associate the contract with both the IdP connection and the SP connection.

When bridging one identity provider to multiple service providers, you need to create an authentication policy contract per service provider because each service provider likely requires a different set of attributes. Map all the authentication policy contracts into the IdP connection. Add the respective authentication policy contract to each SP connection to the service provider.

When bridging multiple identity providers to one service provider, you likely need only one contract unless the service provider requires a different set of attributes from each identity provider. Add the authentication policy contract to the SP connection and the applicable IdP connections.

Authentication policy contracts are managed using the **Policy Contracts** configuration wizard. You can access it from the **Identity Provider** or **Service Provider** menu.

Federation hub and virtual server IDs

PingFederate uses two connections to bridge an identity provider to a service provider:

- 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 identity provider through the IdP connection and generates new assertions to the service provider via the SP connection.

If the SP connection does not use a virtual server ID, the issuer of the assertions (to the service provider) is the ID defined for the protocol between PingFederate (the federation hub as the IdP) and the service provider.

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), for SP-initiated SSO, if the service provider sends AuthnRequest messages to the virtual server ID specific endpoint, PingFederate retains this information automatically. When the identity provider returns the corresponding assertions to PingFederate (the federation hub as the SP), PingFederate retrieves the preserved information and uses that specific virtual server ID as the issuer in the assertions it sends to the service provider. For IdP-initiated SSO, the issuer of the assertions (to the service provider) 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 protocol(s) your deployment will support. For SAML SSO configurations, decide which profiles and bindings will be used. (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). (See *Security infrastructure* on page 115.)

Also, if a stronger signature algorithm is required, determine what RSA algorithm will be used for signing. (The optional algorithm selection is available throughout the administrative console, where signing certificates are specified for various uses.)

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 data stores 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 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 (see *Digital signing policy coordination* on page 117).

Deployment

Decide how PingFederate fits into your existing network. Also, determine whether high-availability, failover options, or both, are required (see the PingFederate *Server Clustering Guide on page 895*).

Federation Server Identification

Determine how you and your partner(s) will identify your respective federation deployments. Under federation standards, both the sender (IdP) and the receiver (SP) of an assertion must be uniquely identified within the identity federation (see *Configuration data exchange* on page 135).

With PingFederate, you define a unique ID for each supported protocol (see *Specifying federation information* on page 137). Optionally, you can also use a list of multiple *Virtual Server IDs* on a connection-by-connection basis (see *Multiple virtual server IDs* on page 133).

ⓘ **Tip:** PingFederate also provides for *virtual host names*, which differ from virtual server IDs (but are not mutually exclusive); they are intended to be used when your network configuration is such that you receive federation messages under more than one domain name (see *Configuring virtual host names* on page 152).

Server Clock Synchronization

Ensure that both the SP and IdP server clocks are synchronized. SAML messages and 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 data store that contains user data when needed (see *Data stores* on page 123).

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.

(See *SSO integration kits and adapters* on page 112 and *Token processors and generators* on page 100.)

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* on page 224.)

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 (see *Identity mapping* on page 119).

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* on page 121.)

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 SP connection to access multiple subdomain accounts in Microsoft Office 365.

Suppose both the marketing and the engineering departments of contoso.com (the 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.

In this scenario, the PingFederate IdP server can be configured to include both marketing.contoso.com and engineering.contoso.com as the virtual server IDs in the Office 365 SP connection. Each virtual server ID has its own set of protocol endpoints, which can be obtained in the connection metadata (see *Metadata export* on page 155 and *System-services endpoints* on page 773 for more information).

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 issuer being 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 (see *Identify the SP* on page 504) or as an SP configuring an IdP connection (see *Identify the partner* on page 599) for both Browser SSO Profiles and WS-Trust STS (for access to identity-enabled web services).

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 default virtual server ID is used when no virtual server ID information is included in a request (see *IdP endpoints* on page 757 as an IdP or *SP endpoints* on page 761 as an SP).

In a connection with multiple virtual server IDs, you can optionally restrict each adapter added to the connection to certain virtual server IDs to enhance the end-user experience (see *Restrict an authentication source to certain virtual server IDs* on page 515 and *Restrict a target session to certain virtual server IDs* on page 608).

ⓘ **Tip:** You can also restrict each token processor or token generator added to a WS-Trust STS SP connection or IdP connection, (see *Restrict a token processor to certain virtual server IDs* on page 685 or *Restrict a token generator to certain virtual server IDs* on page 696).

ⓘ **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 *Define issuance criteria for IdP Browser SSO* on page 519 or *Define issuance criteria for SP Browser SSO* on page 610.

Configuration data exchange

If your partner's deployment does not produce or consume a metadata file that conforms to SAML metadata specifications, you may need to exchange connection information manually. The following sections list some common configuration details that must be exchanged if metadata files are not used. (These lists are not exhaustive.)

IdP to 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 are configuring):

- **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 (see *Federation Server Identification*).
- **SOAP Artifact Resolution URL**—The endpoint your site uses to receive an SP's SOAP requests when the artifact binding is used.
- **Single Logout Service URL**—The destination of SLO request messages.
- **Single Sign-On 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 are configuring):

- **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 (see *Federation Server Identification*).
- **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 parties

Many settings must be mutually set by the parties. This information might include such items as:

- **Attributes**—User information that will be sent in an assertion, if any (see *User attributes* on page 121).
- **Signing certificates**—The SAML and WS-Federation protocols specify a number of conditions under which digital signatures are either required or optional (these conditions are built into the PingFederate connection-setup screens).
- **SOAP connection type** and **authentication style**—For SAML connections using the back channel (using the artifact binding, for example), 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.

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**, and **External Systems**.

Server

The **System**# **Server** menu items let you 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, configure virtual host names, and configure extended properties for connections and OAuth clients.

Protocol settings

On the **System**# **Protocol Settings** screen, you configure the roles and protocols PingFederate plays in your environment, the base URL of your PingFederate environment, the federation identifier of your organization for each enabled protocol, and the optional WS-Trust STS authentication settings if the WS-Trust protocol is enabled.

Choosing roles and protocols

About this task

On the **Roles and Protocols** screen, select the roles your organization plays and the sets of standards you will use with your PingFederate server. Depending on the selected roles and protocols, you may be prompted to provide additional information in a subsequent screen. If your use cases require roles or protocols that have not yet been selected, you must return to this screen to make the selections before you can configure those new use cases.

Steps

1. Go to the **System**# **Protocol Settings**# **Roles & Protocols** screen.
2. Select your federation roles, and then select the applicable protocols.

   Note: Outbound provisioning for SaaS applications requires the use of the SAML 2.0.
3. Optional: If you are using PingFederate as an IdP for provisioning or have installed a SaaS Connector package, select the **Outbound Provisioning** check box.

   If such check box is not available, verify that your PingFederate license includes the **Outbound Provisioning** capability and the outbound provisioning properties are configured in the `<pf_install>/pingfederate/bin/run.properties` file.

   Note: After provisioning is configured for a connection, you cannot clear this check box—you must delete all provisioning configurations first. To suspend provisioning for an SP partner, you can deactivate the specific configuration. Alternatively, you can deactivate the associated SP connection; note, however, that this will also disable SSO/SLO transactions.
4. Optional: If you are using PingFederate as an SP for provisioning, select the **Inbound Provisioning** check box.

5. Optional: If you are using SAML 2.0 XASP as an SP for multiple IdP connections, you may select the option to determine dynamically which connection to use, based on the X.509 certificate presented.

   > **Tip:** After you make this selection and create XASP IdP connections, configure dynamic IdP discovery via the **Attribute Requester Mapping** link on the **Service Provider** menu. Once the mapping is configured, you cannot clear the check box on the **Roles and Protocols** screen unless you first delete the mapping.

   For general information about XASP, see *Attribute Query and XASP* on page 45.
6. Click **Next** and continue with the rest of the configuration.

   > **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Next steps
For information about configuring settings associated with your selections, see these relevant topics:

- *OAuth configuration* on page 378
- *Identity provider SSO configuration* on page 483
- *Outbound provisioning for IdPs* on page 125
- *Service provider SSO configuration* on page 583
- *Provisioning for SPs* on page 126
- *Configuring WS-Trust settings* on page 138
- *Configuring standard IdP Discovery* on page 139

Specifying federation information

About this task

This information identifies your federation deployment to your partners, according to the protocol(s) you support.

> **Note:** You must provide an ID that uniquely identifies your federation gateway for each protocol you support. For WS-Trust STS, IDs are required for both SAML 2.0 and SAML 1.x, regardless of browser-based SSO protocol support or the type of token expected to be issued, to ensure that the STS will perform correctly under all conditions.
>
> 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 (see *Federation planning checklist* on page 132).

Steps

1. Go to the **System# Protocol Settings# Federation Info** screen.
2. Provide the required information.

   For more information, refer to 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 (see *Metadata export* on page 155).

Field	Description
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.

The fields available on this screen depend on the federation protocols enabled on your server (see *Choosing roles and protocols* on page 136).

3. Click **Next** and continue with the rest of the configuration.

> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring WS-Trust settings

About this task

On the **System# Protocol Settings# WS-Trust STS Settings** screen, you may configure PingFederate to require that client applications provide credentials to access the STS.

While this is an optional configuration, it is recommended for 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 may 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 SP partner connections, or token-to-token mappings, using more fine grained controls, at the connection level via Issuance Criteria.

Steps

1. On the **WS-Trust STS Settings** screen, click **Configure WS-Trust STS Authentication** to begin.

   Follow the configuration wizard to complete the task. For more information, see *Configure STS authentication* on page 674
2. Click **Next** and continue with the rest of the configuration.

> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring outbound provisioning settings

About this task

On the **System# Protocol Settings# Outbound Provisioning** screen, select the database that PingFederate should use internally to facilitate provisioning for service providers when PingFederate is configured as an IdP.

The database stores the state of synchronization between the source data store and the target data store, enabling periodic checking to determine whether updates are required at the target site. PingFederate checks the source data store for changes every minute by default. As needed, you may change the provisioning synchronization frequency on this screen as well.

> ⓘ **CAUTION:** A pre-installed, default HSQLDB database is selected for initial setup and testing. However, we strongly recommend that you choose your own, secured database for production deployments.

> ⓘ **Note:** PingFederate has been tested using HSQLDB, Microsoft SQL Server, Oracle Database, Oracle MySQL, and PostgreSQL as internal provisioning data stores. However, any relational database should work as well; adaptable setup scripts used for HSQLDB, Microsoft SQL Server, Oracle Database, Oracle MySQL, and PostgreSQL are provided in the `<pf_install>/pingfederate/server/default/conf/provisioner/sql-scripts` directory.

Note that the **Outbound Provisioning** screen appears only when the **Outbound Provisioning** protocol is enabled on the **System# Protocol Settings# Roles & Protocols** screen.

Steps

1. Select a data store from the **Internal Provisioning Data Store** list.

   If the data store 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 data store.
2. Optional: Change the **Synchronization Frequency** value.

   The default value is `60` (seconds).
3. Click **Next** and continue with the rest of the configuration.

   > ⓘ **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring standard IdP Discovery

About this task

SAML 2.0 IdP Discovery provides a cookie-based look-up mechanism used to identify a user's IdP dynamically during an SP-initiated SSO event, when the IdP is not otherwise specified. This mechanism can be helpful, in particular, in cases where an SP might be a hub for several IdPs in an identity federation.

> ⓘ **Tip:** 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 *Configure IdP discovery using a persistent cookie* on page 672.
>
> Furthermore, 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 may 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. On the **System**# **Protocol Settings**# **Roles & Protocols** screen, select the IdP Discovery role.
2. On the **System**# **Protocol Settings**# **IdP Discovery** screen, click **Configure IdP Discovery**.
3. On the **Domain Cookie Settings** screen, choose the discovery role or roles of your PingFederate server.

   The choices that appear on this screen depend on whether PingFederate is acting as an SP, an IdP, both an SP and an IdP, or an IdP Discovery only server on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
4. On the **Common Domain Service** screen, configure as follows:

Field	Description
Base URL	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** screen, 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 (.); for 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** screen, review and modify settings as needed. Then click **Save**.

   The administrative console brings you back to the **System**# **Protocol Settings**# **IdP Discovery** screen.
7. Click **Next** and continue with the rest of the configuration.

Tip: When editing an existing configuration, you may 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:
   - Ensure 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 in the **Connection Options** screen for the desired SP connection.

Reviewing protocol settings

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Administrative accounts

The PingFederate administrative console supports four authentication schemes:

- Native authentication
- LDAP authentication
- RADIUS authentication
- Certificate-based authentication

For role-based access control, PingFederate provides two account types and three administrative roles, as shown in the following table:

PingFederate User Access Control

Account type	**Administrative role**	**Access privileges**
Admin	Admin	Configure partner connections and most system settings (except the management of local accounts and the handling of local keys and certificates).
Admin	Crypto Admin	Manage local keys and certificates.
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Auditor	*Not applicable*	View-only permissions for all administrative functions. When the **Auditor** role is assigned, no other administrative roles may be set.

Note: All three administrative roles are required to access and make changes through the following services:

- The `/ConfigArchive` administrative API endpoint
- The **System**# **Configuration Archive** screen in the administrative console
- The **Connection Management** configuration item on the **Security**# **Service Authentication** screen

For native authentication, access and authorization are controlled by the local accounts defined on the **System**# **Administrative Accounts** screen.

As needed, you can switch from native authentication to an alternative console authentication. Note that access and authorization are defined in the respective configuration file.

An administrative user may log on from more than one browser or location. Moreover, multiple administrative users can log 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 `<pf_install>/pingfederate/bin/run.properties` file. Regardless of the property configuration, any number of auditors may log 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** screen 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** screen 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 may also single sign-on from the PingOne admin portal to the administrative console.

Enabling native authentication

About this task

When the administrative console is protected by native authentication, access is restricted to the local accounts defined in the **System**# **Administrative Accounts** screen.

Steps

1. In the `<pf_insall>/pingfederate/bin/run.properties` file, change the value of the pf.console.authentication property as shown below:

   ```
   pf.console.authentication=native
   ```
2. Start or restart PingFederate.

Managing local accounts and role assignments

Steps

1. Go to the **System**# **Administrative Accounts** screen.

Task	Steps
Create a local account	**a.** On the **System**# **Administrative Accounts** screen, click **Create User**. **b.** On the **User Information** screen, 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. **c.** On the **Password Generation** screen, 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. **d.** On the **Summary** screen, review your configuration, modify as needed, and then click **Done**. **e.** On the **System**# **Administrative Accounts** screen, select the applicable account type (**Auditor** or **Admin**) and one or more administrative roles for an **Admin** account. **f.** Repeat these steps to create additional accounts.
Modify user information	**a.** On the **System**# **Administrative Accounts** screen, select the account by its username. ⓘ **Note:** Applicable only to active accounts. **b.** On the **User Information** screen, update the record, and then click **Done**. **c.** Repeat these steps to update other accounts.
Update role assignments	**a.** Select a different account type (**Auditor** or **Admin**) for one or more accounts. **b.** 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	**a.** Click **Deactivate** or **Activate** under **Action**. **b.** 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. Click **Save** to keep your configuration.

Enabling notification messages for account management events

About this task

Administrators can optionally enable notifications for account management events. If enabled, PingFederate generates notification messages based on the following events.

Event	Alert
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.
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.
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.

ⓘ **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 the **System# Administrative Accounts** screen.
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. Click **Save** to keep your configuration.

Results

Setting or resetting passwords

About this task

User administrators generate temporary passwords as they create new local accounts for new users in the **Password Generation** screen.

User administrators can also reset and assign temporary passwords for existing users who forget their passwords. Upon successful authentication, the users are required to change their passwords immediately.

> **Note:** If you are using an alternative console authentication for the administrative console or the administrative API, password management is handled by the third-party system.

Steps

1. Go to the **System**# **Administrative Accounts** screen.
2. Optional: Select the **Notify Administrator of Account Change** check box if you want PingFederate to generate a notification message for the administrator whose password is about to be set or reset.
3. Click **Reset Password** under **Action** for the applicable account.
4. On the **Password Generation** screen, enter a password or click **Generate one-time password** to generate a random password for the account, and then click **Save**.

Changing passwords

About this task

Administrative users and auditors can change the passwords of their local accounts at any time.

> **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 the **System**# **Administrative Accounts** screen.
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. Click **Save** to keep your configuration.

License management

When you access the PingFederate administrative console for the first time, you are prompted to import a license file. This applies to new and upgraded PingFederate instances.

Depending on your licensing agreement, your PingFederate license may 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.

Reviewing license information

Steps

1. Start PingFederate and sign on to the administrative console.
2. Point to the user icon in the upper-right corner of the administrative console.
3. Click **About** from the list.

   The license summary is displayed in a pop-up browser window.

   Important: If an expiration date is specified, the license expires at the beginning of the day.

Requesting a new license key

Steps

1. Go to the Ping Identity licensing website.
2. Sign on, provide the required information, and then submit your request.

Results

You shall hear from our licensing team as the team processes your request.

Installing a license key on a new or upgraded PingFederate server

Steps

1. Start PingFederate and sign on to the administrative console.
2. On the **System**# **License Management** screen, click **Choose file** to select the license file and then click **Import**.

   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.

   Once 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 the **System**# **Cluster Management** screen and click **Replicate Configuration**.

   Note: Starting with PingFederate 8.2, you must use the **Replicate Configuration** screen 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: 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 referred to as *the secondary license file* in the previous versions of PingFederate. The `.pingfederate.lic` file, if any, is ignored.

Install a replacement license key

About this task

You may replace your existing PingFederate license key by importing a new license file in the **System**# **License Management** screen.

Steps

1. Start PingFederate and sign on to the administrative console.
2. On the **System**# **License Management** screen, click **Choose File** to select the license file, and then click **Import**.

   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, you are advised and prompted on whether to continue.

   Once 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.
3. If you have a clustered PingFederate environment, go to the **System**# **Cluster Management** screen and click **Replicate Configuration**.

   Note: Starting with PingFederate 8.2, you must use the **Replicate Configuration** screen 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 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, and then click **Replicate Configuration** on the **Cluster Management** screen 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: 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 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

About this task

If your PingFederate license has an expiration date, you may configure PingFederate to notify the administrators ahead of the license expiration date to minimize potential service disruptions.

Steps

1. Go to the **System#** **Runtime Notifications** screen.
2. Select the **Notification for Servers Licensing Events** check box.

   Note that this check box appears only if your PingFederate license has an expiration date.
3. Enter an 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. Click **Save** to keep your configuration.

Configuration archive

PingFederate automatically creates a time-stamped configuration (ZIP) archive every time an administrator signs on to the administrative console and before an existing archive is imported. The archives are stored in the `<pf_install>/pingfederate/server/default/data/archive` directory. These configuration archives can be used as backup files for the current PingFederate installation.

The automatic backup process typically completes without delays. For deployments with hundreds of connections or OAuth clients (or both), administrators can optionally configure PingFederate to create configuration archives periodically instead.

Additionally, administrators can export the current configuration to a ZIP file on the **System#** **Configuration Archive** screen. This screen is only available to administrators, whose accounts have been assigned the User Admin, Admin, and Crypto Admin roles.

CAUTION: As the backup file contains your complete PingFederate configuration, ensure the file is protected with appropriate security controls in place.

On the **System#** **Configuration Archive** screen, administrators can also import an existing archive for immediate deployment into a running PingFederate server.

Furthermore, administrators can deploy a configuration archive manually by copying the ZIP file to `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory.

Configuration archives are intended for administrative-console configuration only. As such, the following files are *not* included in the archives:

- Launch scripts under the `<pf_install>/pingfederate/bin` and `<pf_install>/pingfederate/sbin` directories.
- Web container configuration files under the `<pf_install>/pingfederate/etc` directory.
- Log files under the `<pf_install>/pingfederate/log` directory.
- Database drivers and program files from adapters and any other plug-ins under 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) under the `<pf_install>/pingfederate/server/default/conf` directory.

If any changes have been made to files that are not part of the configuration archive, such files must be preserved manually.

ⓘ **Tip:** You may 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 wish to include them in a full backup archive or in an archive to be used for configuration migration.

Configuring a backup schedule

About this task

For deployments with hundreds of connections or OAuth clients (or both), administrators can optionally configure PingFederate to create configuration archives periodically instead.

Steps

Edit the `org.sourceid.saml20.domain.mgmt.impl.DataArchiveBackup.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Refer to 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 again 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. Note that regardless of the value of this property, PingFederate always create 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 instance, if your deployment has 50 connections, you can certainly 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` on the console node.

Exporting an archive

About this task

Administrators can export the current administrative-console configuration to a ZIP file on the **System**# **Configuration Archive** screen. This screen is only available to administrators, whose accounts have been assigned the User Admin, Admin, and Crypto Admin roles.

Steps

On the **Export** screen, click **Export** and then save the ZIP file.

CAUTION: As the backup file contains your complete PingFederate configuration, ensure the file is protected with appropriate security controls in place.

Importing an archive

About this task

Administrators can import a configuration archive (ZIP) on the **System**# **Configuration Archive** screen. This screen is only available to administrators, whose accounts have been assigned the User Admin, Admin, and Crypto Admin roles.

When an administrator initiates deployment of a configuration archive using the **System**# **Configuration Archive**# **Import** screen, PingFederate displays error messages if there are any missing plug-in components (such as adapters, database drivers, or token translators) on which the archive depends, or any mismatches of PingFederate licensing authorization. However, the administrator can choose to force the deployment and then install the necessary files 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.

Steps

1. On the **Import** screen, choose the desired configuration archive from your system.
2. Select the **Force Import** check box *if* you want PingFederate to deploy the archive regardless of whether dependency errors are detected.

   Important: If you make this selection, consult the server start-up console or the server log for any messages concerning missing plug-in components or other errors.
3. Click **Import**.

   The administrative console will prompt you to confirm the import process.

   To proceed, click **Yes**.

   To abort, click **No** or **Cancel**.

Cluster management

When multiple PingFederate servers are set up to run as a cluster, the administrative console provides a **System**# **Cluster Management** screen.

Whenever applicable changes are made through the administrative console, a message appears at top of the console to serve as a reminder to go to the **Cluster Management** screen and to replicate the current console configuration to all server nodes in the cluster.

Note: Starting with PingFederate 8.2, you must also use the **Replicate Configuration** screen to initiate the transmission of the license file from the console node to all server nodes.

Replicating configuration

Steps

1. On the **System**# **Cluster Management** screen, click **Replicate Configuration**.
2. Click **Done**.

Results

The replication process takes only a moment, during which time any new requests coming into any of the runtime servers are held in a queue. When the replication is complete, processing continues where it left off.

Virtual host names

On the **System**# **Virtual Host Names** screen, you can optionally define a list of alternate domain names at which PingFederate receives application and protocol messages. 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 allows 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 on the **System**# **Protocol Settings**# **Federation Info** screen.

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 via the private_key_jwt client authentication method

An OAuth client may authenticate with an authorization server by presenting a signed JSON Web Token (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 on the **OAuth Server**# **Authorization Server Settings** screen.

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 (see *Multiple virtual server IDs* on page 133). Virtual host names and virtual server IDs are not mutually exclusive. Depending on your use cases and infrastructure, you may 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* on page 293.

Configuring virtual host names

Steps

- To add a new entry, enter the desired value and click **Add**.
- To modify an existing entry, use the **Edit**, **Update**, and **Cancel** workflow.
- To remove an existing entry, use the **Delete** and **Undelete** workflow.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Extended Properties

On the **System# Extended Properties** screen, you can optionally add any number of extended properties to store additional information about connections, OAuth clients, or both. When adding an extended property, you can define it as a single-value property or a multivalued property. Once added, such extended properties become available to all connections and clients. As you create or update a connection or a client, you may 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 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* on page 328.

OAuth attributes fulfillment and issuance criteria

You can use extended properties as attribute sources when fulfilling persistent grants and token contracts. Additionally, you may define issuance criteria to verify extended property values.

Defining extended properties

Steps

1. Go to the **System# Extended Properties** screen.

2. Add any number of extended properties.
   a. Enter the applicable property name under **Name**.

      Once added and saved, the name cannot be modified subsequently. You can however delete the extended property completely.
   b. Optional: Enter a description for this extended property under **Description**.
   c. If this extended property should allow multiple values, select the check box under **Multivalued**.

      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 but 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.
   d. Click **Add**.
   e. Optional: Repeat these steps to define additional extended properties.

      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.

Metadata

You can configure metadata settings, export metadata, or sign metadata XML files, and. These menu items are located under **System# Metadata**.

Metadata settings

On the **System# Metadata Settings** screen, you configure the contact information to be included in your SAML metadata, the metadata signing policy for metadata provided by the `/pf/federation_metadata.ping` federation metadata endpoint, 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

About this task

On the **System# Metadata Settings# System Info** screen, you may provide the contact information to be included in your SAML metadata.

Steps

1. Enter the desired information.
2. Click **Next** and continue with the rest of the configuration.

> ⓘ **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Results

Once configured, 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

About this task

PingFederate generates publicly available metadata for partners through the federation metadata endpoint (`/pf/federation_metadata.ping`). Although optional, it is recommended to sign the metadata, such that partners can verify the authenticity of the metadata.

Steps

1. Go to the **System**# **Metadata Settings**# **Metadata Signing** screen.
2. 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 public key of the metadata signing certificate is included as part of the metadata.
4. Click **Next** and continue with the rest of the configuration.

> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Configuring metadata lifetime

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 serves as an indicator 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 the **System**# **Metadata Settings**# **Metadata Lifetime** screen.
2. To adjust the validity of your metadata, modify the **Cache Duration** field value, in minutes.

   The default value is `1440` (1 day).
3. To adjust the frequency of automatic reloading of SAML metadata, modify the **Reload Delay** field value, in minutes.

   The default value is also `1440` (1 day).
4. Click **Next** and continue with the rest of the configuration.

> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

Reviewing metadata settings

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Metadata export

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.

On the **System**# **Metadata Export** screen, you can export metadata to an XML file by selecting any SAML Browser SSO connection or specifying the desired information manually. The former is also available as a per-connection action item on the **Connections** screen. 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, you 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.

Exporting connection-specific SAML metadata

About this task

You can export metadata for any SAML Browser SSO connection to an XML file. 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 the **System**# **Metadata Export** screen.
2. On the **Metadata Role** screen, select the applicable role.
3. On the **Metadata Mode** screen, select the **Use a connection for metadata generation** option.
   a. 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** screen, select the applicable SAML Browser SSO connection from the list.

   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 may 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.

5. Optional: On the **Metadata Signing** screen, select a certificate to use for signing the metadata XML file.
   a. 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.
   b. Optional: Select the related check boxes to include the public key information and the raw key in the signed XML file.
   c. 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.
6. On the **Export & Summary** screen, click **Export** to save the metadata XML file and then click **Done**.
7. Pass the metadata XML file to your partner.

Exporting selected SAML metadata

About this task

You can manually select the desired information and export a metadata XML file. This is useful for the following situations:

- You have not yet created a SAML Browser 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 the **System**# **Metadata Export** screen.
2. On the **Metadata Role** screen, select the applicable role.
3. On the **Metadata Mode** screen, select the **Select information to include in metadata manually** option.
   a. 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** screen, select the desired version of the SAML protocol from the list.
5. On the **Virtual Host Name** screen, select the applicable virtual host name from the list.

   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** screen, add one or more attributes.

   Modify any entry as needed.

7. Optional: On the **Metadata Signing** screen, select a certificate to use for signing the metadata XML file.
    a. 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.
    b. Optional: Select the related check boxes to include the public key information and the raw key in the signed XML file.
    c. 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.
8. Optional: On the **XML Encryption Certificate** screen, select the certificate that your partner can use to encrypt XML content.

   Applicable only when you have selected **SAML 2.0** on the **Protocol** screen.

   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.
9. On the **Export & Summary** screen, click **Export** to save the metadata XML file and 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 on the **System**# **File Signing** screen.

Signing XML files

Steps

1. On the **System**# **File Signing** screen.
2. On the **Select Metadata File** screen, choose your metadata file.
3. On the **Digital Signature Settings** screen, 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.
    a. Clear the related check boxes to exclude the public key information and the raw key from the signed XML file.
    b. 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.
4. On the **Export & Summary** screen, click **Export** to save the digitally signed file and then click **Done**.

Monitoring and notifications

You can configure PingFederate to process or send SNMP requests from or to your SNMP network-management server. You can also set up notifications for licensing events, certificate events, and SAML metadata update events. These menu items are located under **System**# **Monitoring & Notifications**.

Runtime notifications

On the **System**# **Runtime Notifications** screen, you can configure PingFederate to generate notification messages for various events.

Licensing events

Depending on your licensing agreement, your PingFederate license may have an expiration date. As desired, you can configure PingFederate to generate notification messages when your license is about to expire. Note that 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 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 (see *PingFederate log files* on page 224).

Configuring runtime notifications

Steps

1. Optional: Configure licensing events.
   a. Select the **Notification for Servers Licensing Events** check box.

      Note that this check box appears only if your PingFederate license has an expiration date.
   b. Enter an email address of the intended recipient.
   c. Select a notification publisher instance from the list.

      If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.
2. Optional: Configure certificate events.
   a. Select the **Notification for Certificate Events** check box.
   b. Enter an email address of the intended recipient.
   c. Optional: Enter the initial warning time period (in days) in the **Initial Warning Event** field.
   d. Optional: Enter the final warning time period (in days) in the **Final Warning Event** field.
   e. 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 SAML metadata update events.
   a. Select the **Notification for SAML Metadata Update Events** check box.
   b. Enter an email address of the intended recipient.
   c. Select a notification publisher instance from the list.

      If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

Runtime reporting

PingFederate supports runtime monitoring and reporting through the Simple Network Management Protocol (SNMP), a standard used by network-management consoles to monitor network and server activity across an enterprise. It also supports runtime monitoring and reporting through Java Management Extensions (JMX).

Tip: You can also use HTTP requests at any time to verify the status of the PingFederate server (see *Customize the heartbeat message* on page 290).

Additionally, you may 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 (see *Security audit logging* on page 230).

Configuring SNMP monitoring

About this task

The SNMP management information base (MIB) defines network data available for SNMP monitoring. The MIB file is located in the `<pf_install>/pingfederate/SNMP` directory. The MIB describes the object identifiers that PingFederate uses to communicate information through SNMP. These identifiers are globally unique and managed by the Internet Assigned Numbers Authority (IANA).

SNMP supports *Get*s and *Trap*s.

Gets

A Get is a request for status information sent by a network-management console to an SNMP agent. Embedded within each PingFederate server is an SNMP agent that brokers the communication between the management console and the PingFederate runtime engine.

PingFederate responds to two SSO and SLO types of Get requests:

- The total number of transactions that the server instance has processed since installation
- The total number of failed transactions that the server instance has encountered since installation

In addition, because PingFederate is built within an existing Jetty framework, Gets include a variety of server information available via Jetty-standard Managed Beans (MBeans). A detailed list of this information is provided in the MIB file in the `<pf_install>/pingfederate/SNMP` directory. (For more information about MBeans, see *Runtime monitoring using JMX* on page 160).

Note: If the host operating system does not allow the SNMP agent to bind to privileged ports (those below 1024), consult your operating system's documentation on how to work around this limitation or change the default port 161 to a port above 1023.

Traps

A *Trap* is a spontaneous communication from an agent to a network-management console. PingFederate generates a Trap at regular intervals, the server “heartbeat.” Each Trap contains the amount of time the server instance has been running since its most recent start-up.

Configure access to SNMP monitoring on the **System# Runtime Reporting** screen. As needed, you may configure both Gets and Traps.

Steps

- Optional: Enable Gets.
  a. Select the **Respond to Get Requests** check box.
  b. Modify the **Local Agent Port** and the **Community Name** field values as needed.
- Optional: Enable Traps.
  a. Select the **Generate Traps** check box.
  b. Provide the required information for your network-management console and modify the **"Heartbeat" Interval** field value as needed.
  c. Click **Test SNMP Configuration** to send a single Trap to your network-management console and verify the result.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Runtime monitoring using JMX

Similar to SNMP, 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 (for example, JConsole) may request runtime information from the PingFederate JMX server.

Important: Authentication is required for JMX-client access to PingFederate runtime data (see *Configure service authentication* on page 321).

PingFederate JMX server reports monitoring data for SSO and SLO transactions. In addition, as with SNMP monitoring, numerous Jetty-standard MBeans are available to the PingFederate server's JMX clients.

SSO and SLO monitoring

For SSO/SLO transaction processing, PingFederate provides these MBeans:

- `pingfederate:type=TOTAL_FAILED_TRANSACTIONS`
- `pingfederate:type=TOTAL_TRANSACTIONS`

Each type contains a single attribute, `Count`, which reports the same information as an SNMP `Get`.

Sample Jetty metrics

The following table describes examples of Jetty MBean metrics, available via JMX, that administrators may find useful to supplement information provided via the PingFederate-specific MBeans.

MBean	Attributes
`org.eclipse.jetty.server:connectorstatistics`	`connections` – The total number of TCP connections accepted by the server.
(For Jetty connectors including the primary and secondary PingFederate runtime server ports)	`connectionsDuration`* – How long connections are kept open. Maximum, mean, standard deviation, and total accumulated time are available. `connectionsOpen` – The current number of open connections. Maximum is also available (`connectionsOpenMax`).

MBean	Attributes
`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, etc.
`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`
	Various attributes measuring CPU usage and memory.

Advanced JMX configuration

PingFederate uses port 1099 for its JMX server. To change the port or other JMS configuration items, if needed, modify the `jmx-remote-config.xml` configuration 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 (see *Manage SSL server certificates* on page 293).

External systems

Many use cases require communications between PingFederate and other systems, such as PingOne® or a database server. Underneath **System# External Systems** is where you can set up and maintain such integrations.

Connecting to PingOne for Enterprise after initial setup

About this task

PingOne® for Enterprise is a cloud-based identity as a service (IDaaS) framework for secure identity access management. 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 the **System** menu and select **Connect to PingOne for Enterprise**.
2. Select **Yes, Connect to PingOne for Enterprise** and then click **Sign on to PingOne to get your activation key**.
3. Sign on using your PingOne admin portal credentials.

> **Tip:** If you do not have an account, you are welcome to register for a free trail.

4. Copy the **Activation Key** value from the PingOne admin portal.
5. Close the browser tab and go back to the PingFederate administrative console.
6. On the **PingOne Account** screen, paste the key value in the **Activation Key** field.
7. Click **Next**.

Configuring identity repository settings

About this task

On the **Identities** screen, you can optionally connect to a directory server.

Steps

- To enable directory integration, select **Yes, Connect a Directory Server**.

  You can create a new data store reuse an existing data store in this configuration.

  Create a new data store

  Provide the required information to connect to a directory server and then click **Next**.

  For more information about each field, refer to the following table.

Field	Description
Directory Type	Select the type of the directory server from the list. Refer to *System requirements* on page 60 for a list of supported directory servers.
Data Store Name	Enter the name of the data store.
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 may 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 data store 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** screen appears. If you want to continue without a secure connection, click **Next**. Alternatively, you can go back to the **Identities** screen and specify a different directory server.

If the certificate presented by the directory server is not trusted by PingFederate, the **Certificate Error** screen appears. You can import the certificate used by the directory server to establish a secure connection and then click **NextIdentities** screen and specify a different directory server. . Alternatively, you can go back to the

Use an existing data store

Click **Begin** and then follow the on-screen instructions to create an SP connection to PingOne® for Enterprise.

- 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 POC (proof of concept). Multiple local test accounts are created as a result.

Managing PingOne for Enterprise settings

About this task

On the **System#** **PingOne for Enterprise Settings** screen, configure various PingOne® for Enterprise integration settings and optionally enable and configure a built-in RADIUS server to integrate PingID® with your VPN.

Steps

Configure PingOne for Enterprise settings

- To toggle the ability to sign on to the administrative console using the PingOne admin portal credentials, select or clear the **Enable Single Sign-On from PingOne to the PingFederate Administrative Console** check box.
- To toggle the ability to monitor your PingFederate server (or servers in a clustered environment) from the PingOne admin portal, select or clear the **Enable Monitoring of PingFederate from PingOne** check box.
- To update the authentication key that PingFederate uses to communicate with PingOne for Enterprise, click **Rotate Key**.

  Periodic rotation can ensure optimal security of your environment.

  It is worth noting that PingFederate also automatically rotates the signing certificate used by the managed SP connection. For more information, see *Managed SP connection to PingOne for Enterprise and signing certificate* on page 305.
- To access the PingOne admin portal, click **Launch PingOne Admin Portal**.
- To disconnect PingFederate from your PingOne account, click **Disconnect from PingOne** and then confirm your decision.

  Applicable if you have made changes that *should not be* propagated 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. The test PingFederate 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 will be uploaded to your PingOne for Enterprise account from the test server.

Configure PingOne for Enterprise SSO settings

- To enable SSO via PingOne for Enterprise, click **Identity Repository Configuration**.

  Applicable if you have not yet completed the **PingOne SSO** configuration in the past, which would have created a managed SP connection to PingOne for Enterprise.
- To upload configuration changes to your PingOne account, click **Update PingOne Identity Repository** and then confirm your decision.

  Applicable if you have made changes that *should be* propagated 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. Once 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 becomes available when you set up the new SAML application in PingOne for Enterprise.

Enable and configure the built-in RADIUS server to integrate PingID with your VPN

- Click **PingID Configuration** to open the **PingID VPN (RADIUS)** configuration wizard.

  Applicable if you have not yet completed the **PingID VPN (RADIUS)** configuration in the past, which would have created a PingID provisioning connection, an instance of the PingID PCV, or both.

Configuring SSO from PingOne admin portal to PingFederate administrative console

About this task

In PingFederate 8.0 (or a more recent release), if you have selected to connect PingFederate to PingOne® for Enterprise during the initial setup process, the option to SSO from the PingOne admin portal to the PingFederate administrative console is enabled for you.

Tip: In this scenario, the initial setup does not create any local administrative login. If you decide to disable the SSO from the PingOne admin portal option later, the administrative console will bring you to the **System#** **Administrative Accounts** screen and prompt you to create at least one user with the User Admin role for later use.

If you set up PingFederate without PingOne for Enterprise at the beginning but have subsequently created a managed SP connection to PingOne for Enterprise using the **Connect to PingOne for Enterprise** configuration wizard, you may go to the **System#** **PingOne for Enterprise Settings** screen to enable this option.

Additionally, you can continue to sign on to the administrative console via native or alternative console authentication using the direct login page. You can also disable the direct login 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:
  a. Start a web browser.
  b. Browse to the following 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 reachable from your computer.

  If the SSO option is enabled on the **PingOne for Enterprise Settings** screen and if you have already logged on to the PingOne admin portal, the PingFederate administrative console becomes available. If you are not logged on to the PingOne admin portal, you will be prompted enter your PingOne admin portal credentials. Upon verification, the PingFederate administrative console becomes available.
- To sign on via native or alternative console authentication:
  a. Start a web browser.
  b. Browse to the following URL:

     https://<*pf_host*>:9999/pingfederate/app?service=page/directLogin

     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 reachable from your computer.
- To disable native and alternative console authentication:
  a. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
  b. Change the pf.console.authentication property value to `none`.
  c. Save the change and then restart PingFederate.

  Note: In a clustered PingFederate environment, you are only required to modify the `run.properties` file on the console node.

  After restart, the direct login page is disabled. Administrators can only SSO to the PingFederate administrative console from the PingOne admin portal at https://<*pf_host*>:9999/pingfederate/app.

  If you want to re-enable native or alternative console authentication, update the pf.console.authentication property accordingly and then restart PingFederate.

Monitoring PingFederate from the PingOne admin portal

After establishing a managed SP connection to PingOne® for Enterprise, you can monitor PingFederate from the PingOne admin portal.

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 connected/disconnected status with the date last seen. For each server, you can also drill down for additional information such as CPU load, JVM memory information, and system memory information.

If you do not want to monitor PingFederate using the PingOne admin portal, clear the check box and click **Save** on the **System# PingOne for Enterprise Settings** screen.

Updating the PingOne identity repository

About this task

Once a managed SP connection to PingOne® for Enterprise is established, PingFederate monitors configuration changes that may impact the connection, such as an update to the base URL or an import of a configuration archive that includes a managed SP connection to PingOne for Enterprise. When PingFederate detects such changes, the administrative console prompts you decide whether to update PingOne for Enterprise or to disconnect from PingOne for Enterprise in a banner message.

Steps

- To upload configuration changes to your PingOne for Enterprise account, click **Update Identity Repository**.

  Applicable if you have made changes that *should be* propagated 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. Once 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 becomes available when you set up the new SAML application in PingOne for Enterprise.
- To disconnect PingFederate from your PingOne for Enterprise account, click **PingOne for Enterprise Settings**; then click **Disconnect from PingOne** on the **PingOne for Enterprise Settings** screen.

  Applicable if you have made changes that *should not be* propagated 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. The test PingFederate 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 will be uploaded to your PingOne for Enterprise account from the test server.

Managing data stores

About this task

Data stores represent external systems where user attributes and other data are stored. Once defined, you can configure PingFederate to retrieve user attributes from data stores for contract fulfillment and token authorization in various use cases. You can also configure PingFederate to write certain records or log messages to data stores.

You manage data stores on the **System# Data Stores** screen.

Steps

- To create a new data store, click **Add New Data Store** and then follow the configuration wizard to complete the task.
- To modify an existing data store, select the data store and then follow the configuration wizard to complete the task.
- To review usage of an existing data store, click **Check Usage** under **Action**.
- To remove an existing data store or cancel the removal request, click **Delete** or **Undelete** under **Action**.

  Note: Only data stores that are not currently in-use can be removed.

  As you configure various components on the administrative console, PingFederate performs connectivity tests against the applicable data stores. 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 screen to another.

- To fine-tune caching interval for data-store validation, update the **Data-Store Validation Interval** field value to the desired amount of time (in seconds).

  The default value is `300` in seconds (five minutes).

  A value of `0` turns off the caching and validation tests are executed with each access.

  Note: This setting applies to all data stores.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Results

Adding a new data store

Steps

1. On the **System**# **Data Stores** screen, click **Add New Data Store**.
2. On the **Data Store** screen, enter a name and select the type of the new data store from the list.

   Available types are limited to ones currently installed on your server.
3. To mask attribute values returned from this data store in PingFederate logs, select the **Mask Values in Log** check box.

   This check box is not selected by default.

Configuring a JDBC connection

About this task

On the **Database Config** screen, provide the required information to establish a JDBC connection to your database server.

Note: PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see . To obtain your database driver JAR file, contact your database vendor. Database driver file should be installed to the `<pf_install>/pingfederate/server/default/lib` directory. You must restart the server after installing the driver.

Steps

1. On the **Database Config** screen, configure your JDBC connection.

   For more information about each field, refer to the following table.

Field	Description
Data Store Name	The name of the data store. Applicable only when editing an existing data store.

Field	**Description**
JDBC URL	The location of the database server and the database. The structure of the JDBC URL varies depending on the vendor; for example: ▪ `jdbc:sqlserver://`*`databaseservername`*`;databaseName=`*`databasename`* ▪ `jdbc:oracle:thin:@`*`databaseservername`*`:`*`databasename`* ▪ `jdbc:mysql://`*`databaseservername`*`/`*`databasename`* ▪ `jdbc:postgresql://`*`databaseservername`*`/`*`databasename`* where *`databaseservername`* is the network address of the database server. It can be an IP address, a host name, or a fully qualified domain name. **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`
Driver Class	The name of the driver class used to communicate with the source database; for example: ▪ `com.microsoft.sqlserver.jdbc.SQLServerDriver` ▪ `oracle.jdbc.OracleDriver` ▪ `com.mysql.jdbc.Driver` ▪ `org.postgresql.Driver` The driver class name should be supplied by the database software vendor in a JAR file.
Username	The name that identifies the user when connecting to the database.
Password	The password needed to access the database.
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 may not be able to 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) ▪ `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 data store should be masked in PingFederate logs. Applicable only when editing an existing data store.

Field	Description
Allow Multi-Value Attributes	When selected (the default), indicates that this JDBC data store can select more than one records from a column and return the results as a multivalued attribute. Otherwise, a query returns only the first value in the column.

2. Click **Advanced** to configure additional settings.
   a. On the **Advanced Database Options** screen, 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.

   b. Configure advanced settings.

   For more information about each field, refer to the following table.

Field	Description
Minimum Pool Size	The smallest number of database connections that can remain in the pool for the given data store. A minimum value of `0` means that the number of connections in the pool can be reduced to zero. **Note:** For optimal performance, the value for this setting should be equal to 50% of the maxThreads value in the Jetty server configuration (*Connection pools to data stores* on page 976). Note that PingFederate does not establish the connection pool for the given data store until it receives a request that requires one or more attributes from that data store. Prior to clicking on **Apply Defaults**, this field shows no value and uses the built-in value of `0` from the underlying component. The default value (after clicking on **Apply Defaults**) is `0` as well.
Maximum Pool Size	The largest number of database connections that can remain in the pool for the given data store. **Note:** For optimal performance, the value for this setting should be equal to 75% to 100% of maxThreads value in the Jetty server configuration (see *Connection pools to data stores* on page 976). Prior to clicking on **Apply Defaults**, this field shows no value and uses the built-in value of `8` from the underlying component. The default value (after clicking on **Apply Defaults**) is `20`.
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. Prior to clicking on **Apply Defaults**, this field shows no value and uses the built-in value of `-1` from the underlying component. The default value (after clicking on **Apply Defaults**) is `5000`.

Field	Description
Idle Timeout (min)	The length of time the connections can be 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 data store once the pool is established. Prior to clicking on **Apply Defaults**, this field shows no value and uses the built-in value of `-1` from the underlying component. The default value (after clicking on **Apply Defaults**) is `5`.

3. Click **Save** to keep your configuration.

Configuring an LDAP connection

About this task

On the **LDAP Configuration** screen, provide the required information to establish an LDAP connection to your directory server.

Steps

1. On the **LDAP Configuration** screen, configure your LDAP connection.

For more information, refer to the following table.

Field	Description
Data Store Name	The name of the data store. Applicable only when editing an existing data store.
Hostname(s) (Required)	The network address of the directory server. It can be an IP address, a host name, or a fully qualified domain name. The entry may include a port number; for example, `10.10.10.101:1389`. For failover, you can 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 DN and password (unless the **Bind Anonymously** check box is selected). 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`.

Field	Description
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:** We recommend that all LDAP connections be secured by using LDAPS. **Note:** If you want 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; AD requires this level of security to allow password changes. This check box is not selected by default.
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 may fine-tune the TTL value and the record prefixes on the **Advanced LDAP Options** screen. **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 not selected by default.
LDAP Type (Required)	If you are using this data store for outbound provisioning and your directory server is PingDirectory™ , Microsoft Active Directory, or Oracle Directory Server, select the applicable type from the list , such that PingFederate can pre-populate many provisioning settings on the **Outbound Provisioning**# **Channel**# **Source Settings** screen. **Tip:** If your directory server is *not* one of the three aforementioned directory servers, you may define a custom LDAP Type to streamline the outbound provisioning configuration. The LDAP type is also used to enable password-change messaging between AD and PingFederate when an HTML Form Adapter instance is used.

Field	Description
Bind Anonymously	Select this check box if your directory server supports anonymous binding and if no credentials are needed to access the directory server. When selected, user DN and password are not required. ⓘ **Tip:** For inbound provisioning, because PingFederate needs to manage local user records, your directory server likely requires 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. This check box is not selected by default.
User DN	The username credential required to access the directory server. ⓘ **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 (if applicable) users (and groups if applicable) is required. When connecting to an AD directory server, enter an AD User account; do not use a Computer account. 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 data store. For more information, see *Configuring proxied authorization* on page 175.
Password	The password credential required to access the directory server.
Mask Values in Log	Determines whether all attribute values returned through this data store should be masked in PingFederate logs. Applicable only when editing an existing data store.

2. Optional: Click **Advanced**If you choose an anonymous binding, ensure that to configure additional settings on the **Advanced LDAP Options** screen.
3. Click **Save** to keep your configuration.

Setting advanced LDAP options

About this task

PingFederate maintains a search pool and a bind pool for each LDAP data store for optimal performance. The search pool is meant for LDAP directory searches. The bind pool is meant for LDAP bind authentication purposes. Use the **Advanced LDAP Options** screen 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.

Steps

1. On the **Advanced LDAP Options** screen, 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, refer to the following table.

Field	Description
Test Connection on Borrow	Indicates whether objects are validated before being borrowed from the pool. This check box is not selected by default.
Test Connection on Return	Indicates whether objects are validated before being returned to the pool. This check box is not selected by default.
Create New Connection If Necessary	Indicates whether temporary connections can be created 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 data store instance may fail. This check box is selected by default.
Verify LDAPS Hostname	Indicates whether to verify the hostname of the directory server matches the subject (CN) or one of the subject alternative names (SANs) from the certificate. **Important:** We recommend to verify LDAPS hostname for all LDAPS connections. This check box 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. **Note:** For optimal performance, the value for this setting should be equal to 50% of the maxThreads value in the Jetty server configuration (see *Connection pools to data stores* on page 976). Note that PingFederate does not establish the connection pool for the given data store until it receives a request that requires one or more attributes from that data store. The default value is `10`.

Field	Description
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** check box is selected). The value must be greater than or equal to the **Minimum Connections** value. **Note:** For optimal performance, the value for this setting should be equal to 75% to 100% of maxThreads value in the Jetty server configuration (see *Connection pools to data stores* on page 976). The default value is `100`.
Maximum Wait (Milli) (Required)	The maximum number of milliseconds the pool waits for a connection to become available 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 be returned 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 should be allowed to 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`.

3. Optional: Click **Next** to specify LDAP binary attributes in the **LDAP Binary Attributes** screen.
4. Click **Save** to keep your configuration.

Specifying LDAP binary attributes

About this task

PingFederate allows you to specify attributes where such attribute values must be handled as binary data for use in attribute contract fulfillment.

ⓘ **Note:** Binary data cannot be used in an assertion. Encoding must be applied and is handled 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 **LDAP Binary Attributes** screen, add, edit, or remove binary attributes.
2. Click **Save** to keep your configuration.

Configuring proxied authorization

About this task

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 data store. By doing so, users are not allowed to reset their passwords if their accounts have been disabled or if they have not been granted the permission to change their passwords.

Refer to the following resources to configure proxied authorization for the service account.

Steps

- For PingDirectory, see *Working with Proxied Authorization* in the *PingDirectory Administration Guide*.
- For Oracle Directory Server, see the *Oracle Fusion Middleware Deployment Planning Guide* (docs.oracle.com/cd/E29127_01/doc.111170/e28974/security-requirements.htm#aalhm) and search for *Proxy Authorization*.

Results

Note that Microsoft Active Directory does not support proxied authorization (see the *Microsoft Active Directory Technical Specification* at msdn.microsoft.com/library/cc223358.aspx).

For general information about proxied authorization, please refer to *RFC4370*.

Configuring the account usability control ACI

About this task

When connecting to PingDirectory™ , configure the account usability control ACI for the service account on the directory server if you intend to enable self-service account unlock in any HTML Form Adapter instances that use this data store.

Steps

1. Create an LDIF file to capture the following ACI information.

   OID

   1.3.6.1.4.1.42.2.27.9.5.8

   Name

Account Usability Control

Permission

read

Suppose you create the following file and name it `aci.ldif`.

```
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetcontrol="1.3.6.1.4.1.42.2.27.9.5.8")(version 3.0; acl "Access
 to the Account Usability Control"; allow (read) userdn="ldap:///
uid=ServiceAccount,ou=Applications,dc=example,dc=com";)
```

2. Use the **`ldapmodify`** command to configure the required ACI.

```
$ ldapmodify -f <path>/aci.ldif
-h <host name>
-p <LDAP port>
-D <LDAP bind username>
-w <LDAP bind password>
```

(Line breaks are inserted for readability only.)

Configuring the password validation details request control ACI

When connecting to PingDirectory, configure the password validation details request control 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 this data store.

About this task

For self-service password management—the scenario where the user knows the current password and wants to update it—the service account of this data store must be equipped with the password validation details request control ACI. For self-service account recovery—the scenario where the users 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

1.3.6.1.4.1.30221.2.5.40

Name

Password Validation Details Request Control

Permission

read

Suppose you create the following files for change password and password reset:

`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";)
```

`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.

```
$ 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>
```

Line breaks are inserted for readability only.

Defining a custom LDAP type for outbound provisioning

About this task

If you are using outbound provisioning and your directory server is not PingDirectory™ , Microsoft Active 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 the `sample.template.txt` file located in the `<pf_install>/pingfederate/server/default/conf/template/ldap-templates` directory.
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** screen 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 the **Outbound Provisioning# Channel# Source Settings** screen. They help the provisioner to 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.

Results

Once configured, you may create a new LDAP data store using the newly defined LDAP type. To streamline outbound provisioning configuration, select the LDAP data store that uses the newly defined LDAP type on the **Source** screen.

Configuring other types of data stores

About this task

Beside connecting to a directory server via LDAP or a database server via JDBC, PingFederate is also capable of connecting to other types of data stores, such as REST API-enabled data source that returns user attributes in JavaScript Object Notation (JSON).

Steps

- Refer to subsequent topics for configuration steps.

Configuring a REST API data store

About this task

To retrieve attribute data from a JSON-based REST API, you must first create a REST API data store using the **System**# **Data Stores** configuration wizard. On the **Configure Data Store Instance** screen, provide the required information to set up the data store.

Steps

1. If the data source requires specific HTTP request headers, define them under **HTTP Request Headers**.

If configured, PingFederate includes the configured HTTP request headers (and their values) when contacting the data source.

a. Click **Add a new row to 'HTTP Request Headers'**.
b. Enter the applicable name and value under **Header Name** and **Header Value**.
c. Click **Update** under **Action**.

Repeat these steps to define additional HTTP request headers and their values.

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.

2. Define local attribute names and map them to the data return by the data source under **Attributes**.

   You 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 (tools.ietf.org/html/rfc6901).

   You must define at least one attribute.

   a. Click **Add a new row to 'Attributes'**.
   b. Enter the local attribute name and path under **Local Attribute** and **JSON Response Attribute Path**.
   c. Click **Update** under **Action**.

   Repeat these steps to define additional attributes.

   Tip: Define only the attributes that are required by other configuration items, such as contract fulfillment or token authorization. Provide meaningful attribute names so that they can be easily recognized at a later time.

   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.
3. Enter the base URL of the data source offering REST API access to its data.

   In the event that 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 *Specify a resource path for a REST API data store* on page 877.
4. Configure the authentication method.

   For more information, refer to the following table.

Authentication method	Required configuration
None	PingFederate makes unauthenticated REST API requests to the data source. No credential information is required. This is the default selection.
Basic Authentication	PingFederate authenticates via the HTTP Basic authentication scheme. Enter the required credentials in the **Username** and **Password** fields.
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.

5. If PingFederate should mask attribute values returned through this data store in its log, select the **Mask Values in Log** check box.

   Applicable only when editing an existing data store.

   This check box is not selected by default.

6. Optional: Click **Advanced** to configure additional settings.

   For more information, refer to the following table.

Field	Description
Enable HTTPS Hostname Verification	Indicates whether to verify the hostname of the data source matches the subject (CN) or one of the subject alternative names (SANs) from the certificate. **Important:** We recommend to verify hostname for all connections. This check box is selected by default.
Read Timeout	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	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 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 may retry the request. For example, you may 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 GET requests to test the data store connection on the **Actions** screen. When not specified (the default), PingFederate sends GET requests to the base URL of the data store.

7. On the **Actions** screen, verify the data store configuration.
    a. Click **Test Connection** to test the connectivity between PingFederate and the data source.

       The administrative console displays the results returned by the data source. The PingFederate server log may contain additional messages as well.
    b. Review the results.
    c. Optional: Click **Reset** and repeat the test again.
8. On the **Summary** screen, 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; for example:

```
{
  "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 data store 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.

Configuring a custom data store

About this task

Developers can use the PingFederate SDK to create specific drivers for non-JDBC or LDAP data stores (or more sophisticated JDBC or LDAP queries) including; for example, flat files or SOAP-connected databases. Furthermore, data stores may be written to perform configuration assistance or validation actions, such as testing a connection to a database. Actions may also include generation of parameters that might need to be set manually in a configuration file. Once the data-store driver (JAR) file is written and installed, you can select it on the **Data Store** screen when creating a new instance of your data store.

For more information, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.

Steps

1. On the **Configure Data Store Instance** screen, configure your data store instance.

   Depending on the data store implementation, configuration requirements vary.

   For example, after building and deploying the sample from the `<pf_instal>/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 illustrated in the following screen captures.

   When editing an existing instance, you can optionally modify the name of the data store instance and toggle the option whether PingFederate should mask attribute values returned from this data store instance in PingFederate logs.
2. On the **Actions** screen (if shown), follow the on-screen instructions provided by the developer to complete any required tasks.

   Depending on the data store implementation, configuration requirements vary. If no action is required, this screen is not shown.
3. Click **Save** to keep your configuration.

Defining a data store for persistent authentication sessions

About this task

When enabling PingFederate authentication sessions, you can optionally select the persistent option so that PingFederate can leverage previous sessions as users request protected resources after restarting their browsers. This optional configuration requires session-state data to be stored externally, as opposed to in-memory alone. By default, PingFederate uses its internal HSQLDB database to maintain persistent authentications. You can also configure PingFederate to maintain persistent authentication sessions externally on a database server or a PingDirectory™ server.

Important: When persistent authentication sessions are expected, consider maintaining session-data securely on an external storage medium for production standalone deployments.

For server clustering, an external storage is required because the internal HSQLDB database cannot be shared across other PingFederate engine nodes.

Changing the default storage involves two tasks.

Steps

1. Create the required data structure on the external storage medium.
2. Modify two PingFederate configuration XML files.

Configuring an external database for authentication sessions

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 for your database server provided in the `<pf_install>/pingfederate/server/default/conf/authentication-session/sql-scripts` directory.
2. If you have not already done so, create a JDBC data store for your database server on the **System#** **Data Stores** screen.
3. Copy the system ID of the applicable JDBC data store from the **System#** **Data Stores** screen.
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 console node.

    Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

    `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 For example, if the system ID is `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.
    a. Locate the `SessionStorageManager` service point:

    ```
    <!-- Service for storing Authentication Sessions. -->
    <service-point id="SessionStorageManager"
     interface="org.sourceid.saml20.service.session.data.SessionStorageManager">
        <invoke-factory>
            ...
            <construct
     class="org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
    >
        </invoke-factory>
    </service-point>
    ```

    b. Set the value of the class attribute to `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl` (the default value).
    c. Save the file.

    Note: For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

6. Start or restart the PingFederate service.

    Note: For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System#** **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Results

For example, if the system ID isPingFederate removes expired authentication sessions from the database once a day. To fine-tune the frequency and the number of expired authentication sessions to be removed, see *Managing authentication sessions stored in the database* on page 278.

Configuring PingDirectory for authentication sessions

About this task

Specific schema objects are required in order for PingFederate to store authentication sessions on your directory server. LDIF scripts are provided for PingDirectory™ .

Steps

1. Update the LDAP schema.
   a. Sign on to the PingDirectory administrative console.
   b. Go to the **LDAP Schema# Schema Utilities** screen.
   c. Click **Import Schema Element**.
   d. 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 placeholder values with relevant information from your directory server.
   e. 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

Use PingDirectory's **`dsconfig`** utility to create these indexes. The **`dsconfig`** utility is interactive. You can also provide inputs as command arguments. For example, the following samples create the pf-authn-session-group-hashed-session-id and pf-authn-session-group-expiry-time 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-expiry-time \
  --set index-type:ordering
```

After adding the indexes, use the **`rebuild-index`** utility to build the indexes. For instance, the following sample builds the required indexes.

```
$ 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 the **System# Data Stores** screen.
4. Copy the system ID of the applicable LDAP data store from the **System# Data Stores** screen.
5. Edit the `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl.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 console node.

   a. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

      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>
...
```

   b. 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.

   c. 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.
   d. Save the file.

6. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.
   a. Locate the `SessionStorageManager` service point:

```
<!-- Service for storing Authentication Sessions. -->
<service-point id="SessionStorageManager"
 interface="org.sourceid.saml20.service.session.data.SessionStorageManager">
    <invoke-factory>
        <!--
        Supported classes are

 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 a LDAP implementation.
        -->
        <construct
 class="org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl
>
    </invoke-factory>
```

```
</service-point>
```

b. Set the value of the class attribute to `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerLdapImpl`.
c. Save the file.

> ⓘ **Note:** For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

7. Start or restart the PingFederate service.

> ⓘ **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System#** **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Results

When storing persistent authentication sessions on a PingDirectory server, you must also configure a cleanup plug-in in PingDirectory to remove expired authentication sessions from your directory server. For more information, see *Managing authentication sessions stored in PingDirectory* on page 279.

Defining an OAuth grant data store

About this task

PingFederate uses a pre-installed HSQLDB database as its persistent grant data store after the initial setup. Authorization grants obtained by OAuth clients in the following manners are considered persistent.

- Grants obtained or updated by using the **Authorization Code**, **Resource Owner Credentials**, or **Device Authorization** grant type, in conjunction with the **Refresh Token** grant type.

  If the use cases involve mapping attributes from authentication sources (IdP adapter instances or IdP connections) or Password Credential Validator (PCV) instances to the access tokens (directly or through persistent grant extended attributes), such attributes and their values are stored along with the persistent grants so that they can be reused 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.

  If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens (directly or through persistent grant extended attributes), attribute values are obtained at runtime for each token request. No attributes or their values are stored with the persistent grants.

Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked or cleaned up.

We strongly recommend to use a secured external storage for production deployments. Supported storage platforms database server, directory servers, and other storage solutions through the use of the PingFederate SDK. An external storage may also address performance or scalability requirements that exceed the HSQLDB database's capabilities.

> ⓘ **Important:** For server clustering, an external grant storage is required because the internal HSQLDB database cannot be shared across other PingFederate engine nodes.

> ⓘ **Note:** Attribute values are always stored encrypted when a directory is used. If a database server is used (including the internal HSQLDB database), attribute values are also stored encrypted by default.

Changing the default storage involves two tasks.

Steps

1. Create the required data structure on the external storage medium.
2. Modify two PingFederate configuration XML files.

Configuring an external database for grant storage

About this task

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.

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 the **System# Data Stores** screen.
3. Copy the system ID of the applicable JDBC data store from the **System# Data Stores** screen.
4. Edit the `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.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 console node.

   Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

   For 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/META-INF/hivemodule.xml` file.
   a. Locate the `AccessGrantManager` service point:

```
<!-- Service for storage of access grants -->
<service-point id="AccessGrantManager"
 interface="com.pingidentity.sdk.accessgrant.AccessGrantManager">
    ...
    <create-instance
 class="org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl"/>
</service-point>
```

   b. Set the value of the class attribute to `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl` (the default value).
   c. Save the file.

Note: For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

6. Start or restart the PingFederate service.

> **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System# Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Results

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* on page 282.

Configuring a directory for grant storage

About this task

Specific schema objects are required in order for PingFederate 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.

4. If you have not already done so, create an LDAP data store for your directory server on the **System# Data Stores** screen.
5. Copy the system ID of the applicable LDAP data store from the **System# Data Stores** screen.
6. Edit the configuration file relevant to your directory server.

   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.AccessGrantManagerLDAPPingDirecto`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl.xml`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl.`

   > **Note:** For a clustered PingFederate environment, edit this file on the console node.

   a. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

      For 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>
...
```

b. Enter a value for the `<c:item name="SearchBase"/>` element.

> **Tip:** This is the distinguished name (DN) that points to the access grants location. For more information, see the inline comment and the LDIF scripts in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.

c. 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.

d. Save the file.

7. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.

a. Locate the `AccessGrantManager` service point:

```
<!-- Service for storage of access grants -->
<service-point id="AccessGrantManager"
 interface="com.pingidentity.sdk.accessgrant.AccessGrantManager">
     <create-instance
 class="org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl"/>
</service-point>
```

b. Update the class attribute value to one of the following values:

Directory server	Class value
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPPingDirect`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl`

c. Save the file.

> **Note:** For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

8. Start or restart the PingFederate service.

> **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System#** **Cluster Management** screen; then 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 PingDirectory, see *Granting storage performance considerations* on page 190 for more information.

Attribute name	Index type
`accessGrantGuid`	equality
`accessGrantUniqueUserIdentifier`	equality
`accessGrantHashedRefreshTokenValue`	equality
`accessGrantClientId`	equality
`accessGrantExpires`	ordering

Results

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* on page 282.

Granting storage performance considerations

If you use PingDirectory™ , or another directory, to store OAuth persistent grants for PingFederate, the following attributes *must* be indexed to ensure that access grant queries perform efficiently.

Attribute name	**Index type**
accessGrantGuid	equality
accessGrantUniqueUserIdentifier	equality
accessGrantHashedRefreshTokenValue	equality
accessGrantClientId	equality
accessGrantExpires	ordering

Use PingDirectory's **`dsconfig`** utility to create these indexes. The **`dsconfig`** utility is interactive. You can also provide inputs as 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
```

After adding the indexes, use the **`rebuild-index`** utility to build the indexes. For instance, 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
```

For more information, see Working with Indexes in the *PingDirectory Administration Guide* .

Furthermore, you may configure a PingDirectory plug-in to handle the cleanup of expired persistent grants and the associated attributes. The plug-in allows fine-grained control over various aspects of the cleanup task, which could smooth out the performance impact. For more information, see *Manage expired persistent grants in PingDirectory* on page 283.

Using a custom solution for grant storage

About this task

You may use the PingFederate SDK to implement a custom solution for grant storage.

Steps

1. Implement the `AccessGrantManager` interface.

   For more information, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.

2. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.
   a. Locate the `AccessGrantManager` service point:

      ```
      <!-- Service for storage of access grants -->
      <service-point id="AccessGrantManager"
       interface="com.pingidentity.sdk.accessgrant.AccessGrantManager">
           <create-instance
       class="org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl"/>
      </service-point>
      ```
   b. Update the class attribute value to the name of your class.
   c. Save the file.

   > **Note:** For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

3. Deploy the required program files of your custom implementation to all PingFederate servers.
4. Start or restart the PingFederate service.

   > **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System# Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Defining an OAuth client data store

About this task

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 provides the flexibility to manage client records via 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. Instead, they are stored on a database server, a directory server, or some other storage medium through the use of the PingFederate SDK.

Changing the default storage involves two tasks.

Steps

1. Create the required data structure on the external storage medium.
2. Modify two PingFederate configuration XML files.

Configuring an external database for client storage

About this task

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.

> **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, you may 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 .

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.
   a. Locate the `ClientManager` service point:

```
<!-- Service for storing OAuth client configuration. -->
<service-point id="ClientManager"
 interface="org.sourceid.oauth20.domain.ClientManager">
    <invoke-factory>
        <!--
        Supported classes are
        org.sourceid.oauth20.domain.ClientManagerXmlFileImpl ...
        org.sourceid.oauth20.domain.ClientManagerJdbcImpl    ...
        org.sourceid.oauth20.domain.ClientManagerLdapImpl    ...
        org.sourceid.oauth20.domain.ClientManagerGenericImpl ...
        -->
        <construct
 class="org.sourceid.oauth20.domain.ClientManagerXmlFileImpl"/>
    </invoke-factory>
```

```
</service-point>
```

   b. Update the class attribute value to `org.sourceid.oauth20.domain.ClientManagerJdbcImpl`.
   c. Save the file.

ⓘ **Important:** For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

Additionally, you must set up an external database because the bundled HSQLDB database cannot be shared across multiple PingFederate engine nodes. For production standalone deployments, it is also recommended to store the client records in an external secured database.

Follow the remaining steps to set up an external database for client storage.

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 JDBC data store for your database server on the **System# Data Stores** screen.
4. Copy the system ID of the applicable JDBC data store from the **System# Data Stores** screen.
5. Edit the `org.sourceid.oauth20.domain.ClientManagerJdbcImpl.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 console node.

Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

For 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 the PingFederate service.

ⓘ **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System# Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Configuring a directory for client storage

About this task

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.

ⓘ **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, you may 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 .

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.
4. If you have not already done so, create an LDAP data store for your directory server on the **System# Data Stores** screen.
5. Copy the system ID of the applicable LDAP data store from the **System# Data Stores** screen.
6. Edit the `org.sourceid.oauth20.domain.ClientManagerLdapImpl.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 console node.

   a. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

      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>
      ...
      ```
   b. 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.
   c. 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.
   d. Save the file.
7. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.

   a. Locate the `ClientManager` service point:

      ```
      <!-- Service for storing OAuth client configuration. -->
      <service-point id="ClientManager"
       interface="org.sourceid.oauth20.domain.ClientManager">
          <invoke-factory>
              <!--
              Supported classes are
              org.sourceid.oauth20.domain.ClientManagerXmlFileImpl ...
      ```

```
         org.sourceid.oauth20.domain.ClientManagerJdbcImpl    ...
         org.sourceid.oauth20.domain.ClientManagerLdapImpl    ...
         org.sourceid.oauth20.domain.ClientManagerGenericImpl ...
         -->
         <construct
  class="org.sourceid.oauth20.domain.ClientManagerXmlFileImpl"/>
      </invoke-factory>
 </service-point>
```

b. Update the class attribute value to `org.sourceid.oauth20.domain.ClientManagerLdapImpl`.
c. Save the file.

Note: For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

8. Start or restart the PingFederate service.

Note: For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System# Cluster Management** screen; then 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 PingDirectory, see *Client storage performance considerations* on page 195 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`	equity
`pf-oauth-client-name`	ordering
`pf-oauth-client-name`	substring
`pf-oauth-client-last-modified`	ordering

Client storage performance considerations

If you use PingDirectory™ , or another directory, to store OAuth client records for PingFederate, the following attributes *must* be indexed.

Attribute name	Index type
pf-oauth-client-id	equality
pf-oauth-client-id	ordering
pf-oauth-client-id	substring
pf-oauth-client-name	equity
pf-oauth-client-name	ordering
pf-oauth-client-name	substring
pf-oauth-client-last-modified	ordering

Use PingDirectory's **`dsconfig`** utility to create these indexes. The **`dsconfig`** utility is interactive. You can also provide inputs as command arguments. For example, the following sample 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
```

After adding the indexes, use the **`rebuild-index`** utility to build the indexes. For instance, the following sample 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
```

For more information, see Working with Indexes in the *PingDirectory Administration Guide* .

Using custom storage for OAuth clients

About this task

You may use the PingFederate SDK to implement a custom solution for client storage.

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, you may 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 .

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, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.

2. Edit the `<pf_install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` file.

   a. Locate the `ClientStorageManager` service point:

```
<!-- Service for storing OAuth client configuration. -->
<service-point id="ClientManager"
 interface="org.sourceid.oauth20.domain.ClientManager">
    <invoke-factory>
        <!--
        Supported classes are
        org.sourceid.oauth20.domain.ClientManagerXmlFileImpl ...
        org.sourceid.oauth20.domain.ClientManagerJdbcImpl    ...
        org.sourceid.oauth20.domain.ClientManagerLdapImpl    ...
        org.sourceid.oauth20.domain.ClientManagerGenericImpl ...
        -->
        <construct
 class="org.sourceid.oauth20.domain.ClientManagerXmlFileImpl"/>
    </invoke-factory>
</service-point>
```

b. Update the class attribute value with the name of the class implementing the `ClientStorageManagerV2` interface.
c. Save the file.

> **Note:** For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

3. Start or restart the PingFederate service.

> **Note:** For a clustered PingFederate environment, replicate this new configuration to other engine nodes on the **System**# **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Defining an account-linking data store

About this task

When an SP is configured to use account linking for an IdP connection, PingFederate uses an embedded HSQLDB database as the account-link repository. This default implementation does not require any changes to PingFederate to support account linking in a standalone environment. You can also configure PingFederate to store account links on a database server or a directory server. An external storage may also address performance or scalability requirements that exceed the HSQLDB database's capabilities. It can also address the scenario where you and your federation partner previously established a different system for creating and mapping opaque pseudonyms, and PingFederate needs access to the system.

> **Important:** For server clustering, an external grant storage is required because the internal HSQLDB database cannot be shared across other PingFederate engine nodes.
>
> For production standalone deployments, consider maintaining account links securely on an external storage medium.

Changing the default storage involves two tasks.

Steps

1. Create the required data structure on the external storage medium.
2. Modify two PingFederate configuration XML files.

Configuring an external database server for account linking

About this task

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 by using one of the table-setup scripts located in the `<pf_install>/pingfederate/server/default/conf/account-linking/sql-scripts` directory.
2. On the **System**# **Data Stores** screen, create a new data store to connect PingFederate to the database (see *Configuring a JDBC connection* on page 167).

3. On the **System**# **Data Stores** screen, copy the system ID of the new accounting-linking data store.
4. Edit the `org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl.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 console node.

   Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection and save the file.

   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 the **System**# **Cluster Management** screen and replicate this change to other runtime servers.

Configuring a directory server for account linking

Before you begin

Note: User accounts to be linked must exist in the directory prior to establishing the account link. The Account Linking Service does not add users to the directory server but simply updates `AccountLinkDataAttribute` for a given user.

Steps

1. On the **System**# **Data Stores** screen, create a new data store to connect PingFederate to the database (see *Configuring an LDAP connection* on page 170).
2. On the **System**# **Data Stores** screen, copy the system ID of the new accounting-linking data store.
3. Edit the `hivemodule.xml` file.

   The file is located in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.

   Locate the `service-point` for the Account Linking Service; for example:

```
...
<!-- Service/adapter for storage of account linking -->
<service-point id="AccountLinkingService"
 interface="org.sourceid.saml20.service.AccountLinkingService">
    <!--
            Supported classes are

 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
    -->
    <create-instance
 class="org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl"/>
```

```
</service-point>
...
```

Update the class value to `org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl`; for example:

```
...
<!-- Service/adapter for storage of account linking -->
<service-point id="AccountLinkingService"
 interface="org.sourceid.saml20.service.AccountLinkingService">
    ...
    <create-instance
 class="org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl"/>
</service-point>
...
```

Locate the Service-Point ID for AccountLinkingService and change the value of the `create-instance class` to:

`org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl`

4. Edit the `org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl.xml` file.

   The file is located in the `<pf_install>/pingfederate/server/default/data/config-store` directory. The default content of the file reads:

```
<?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 applicable values between the XML tags as follows:

Item name	Element value
PingFederateDSJNDIName	The system ID of new account-linking data store.
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. See this *article* from Microsoft for further information on extending the Active Directory schema (msdn.microsoft.com/library/ms676900(v=VS.85).aspx).

5. Start or restart PingFederate.

6. If you are running PingFederate in a cluster, go to the **System# Cluster Management** screen and replicate this change to other runtime servers.

   ⓘ **Note:** You must also manually apply the changes made in the `hivemodule.xml` file on each runtime server 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`.
   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>
```

Managing Password Credential Validator instances

About this task

PingFederate provides an authentication mechanism using plug-in password credential validators (PCVs). This feature provides centralized credential validation for various PingFederate components and configurations.

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 attribute value (or values) from the applicable PCV instance (or instances). Note that you can only create one grant-mapping configuration per applicable PCV instance.

Finally, 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 plug-in PCVs.

LDAP Username Password Credential Validator

Validates credentials based on an LDAP look-up in an organization's user-data store.

PingID PCV (with integrated RADIUS server)

Validates credentials from a VPN RADIUS client based on an LDAP look-up in an organization's user-data store.

(For more information, see *Integrate PingID with your VPN*.)

PingOne Directory Password Credential Validator

Validates credentials stored in PingOne® 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.

You manage PCV instances on the **System**# **Password Credential Validators** screen.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Results

Note: Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.

If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

Choosing a Password Credential Validator

About this task

The first step in this configuration is choosing the type of the Password Credential Validator. Available types are determined by plug-in JAR files loaded in the `<pf_install>/pingfederate/server/default/deploy` directory. Several validator plug-ins are bundled with PingFederate. Other plug-ins may be added periodically, available from the Ping Identity Downloads website.

Steps

1. Enter a name and an ID for the instance on the **Type** screen.
2. Select the type of the PCV from the list.
3. Optional: 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 screens.

Configuring a Password Credential Validator instance

About this task

The instance configuration of a Password Credential Validator (PCV) varies depending on the credential validators deployed on your server. For PCVs bundled with PingFederate, refer to one of the following topics:

- *Configuring the LDAP Username Password Credential Validator* on page 202

- *Configuring the PingOne Directory Password Credential Validator* on page 206
- *Configuring the RADIUS Username Password Credential Validator* on page 207
- *Configuring the Simple Username Password Credential Validator* on page 208

Configuring the LDAP Username Password Credential Validator

About this task

The LDAP Username Password Credential Validator (PCV) verifies credentials using an organization's LDAP data store.

When an authentication error occurs, PingFederate automatically parses the messages returned by PingDirectory™ , Microsoft Active Directory (AD), or Oracle Directory Server (ODS) and categorize them with the following error conditions:

- Account disabled
- Account expired
- Account locked
- Attribute value invalid
- Attribute conflict
- Invalid credentials
- Invalid telephone number
- Not permitted to logon at this time
- Not permitted to logon at this workstation
- Password expired
- Password policy violated
- Please try again later
- User already exists
- User must reset password
- User not found

As needed, and when validating against a directory server other than PingDirectory, AD, or ODS, administrators can define custom message categorization by mapping specific error messages (with wildcard support) to the desired error conditions on the **Instance Configuration** screen.

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 may create custom experiences based on the error responses, which contain the error messages. The HTML Form Adapter also 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.

As needed, you may localize these messages by using the PingFederate localization framework for an international audience (see).

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

Steps

1. Optional: Override authentication error messages.

   > ⓘ **Note:** This option may be required for a directory server other than PingDirectory, AD, or ODS to support the password change function in the HTML Form Adapter or to alter the end-user messages associated with that function.

   a. Click **Add a new row to 'Authentication Error Overrides'**.
   b. Enter an applicable LDAP error message under **Match Expression**.

      You may use wildcard asterisks (`*`) to match messages returned from your directory server; for example: `*expired*`
   c. Select a relevant error condition from the **Error** list.
   d. Click **Update** under **Action**.
   e. Repeat these steps to add more overrides as needed.

   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.

   Use the up and down arrows to change their display order. The display order does not affect runtime processing.
2. Select the LDAP data store and enter information into the required fields.

   For more information about each field, refer to the following table:

Field	Description
LDAP Datastore (Required)	The LDAP data store 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 data store connection to your AD LDAP server using LDAPS; AD requires this level of security to allow password changes. There is no default selection.
Search Base (Required)	The location in the directory server from which the search begins. This field has no default value.

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 may 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 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: ▪ You can extend the PCV contract and fulfill the subject identifier through the HTML Form Adapter (see *Extending the contract for the credential validator* on page 209). ▪ You can add a data source in the SP connection and fulfill the subject identifier through a data store query (see *Configure attribute sources and user lookup* on page 516). ▪ 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 (see *Applying policy contracts or identity profiles to authentication policies* on page 349). Similarly, 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 via an instance of the Composite Adapter or an authentication policy. For more information, see **Input User ID Mapping** in *Configure a Composite Adapter instance* on page 740 or **Incoming User ID** in *Specifying an incoming user ID* on page 344, respectively. 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 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 screen. This check box is selected by default.

Field	Description
Advanced fields for self-service password reset, account unlock, and username 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 username 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 username 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 username 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. This field has no default value.
Mail Search Filter (for username recovery)	The LDAP query to locate a user record using an email address; for example: `mail=${mail}` This field is required when username 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. This field has no default value.

Field	**Description**
Username Attribute (for username recovery)	The LDAP attribute containing the user identifier of the users. This field is required when username 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. This field has no default value.
Mail Verified Attribute (for username recovery)	The LDAP attribute indicating whether the user's email address has been verified. The expected value of this user attribute must either be `true` or `false` (case insensitive). This field is required when username recovery using only verified email addresses is enabled in any HTML Form Adapter instances that validate credentials against this LDAP Username PCV instance. This field has no default value.

Configuring the PingOne Directory Password Credential Validator

About this task

The PingOne® Directory Username Password Credential Validator verifies credentials stored in the PingOne Directory.

ⓘ **Note:** The PingOne Directory Password Credential Validator requires a PingOne for Enterprise account. For more information, see *Manage PingOne Directory Users* in the *PingOne for Enterprise Administration Guide*.

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

Steps

Enter your account information in the **Client ID** and **Client Secret**.

For more information about each field, refer to the following table. All fields are required.

Field	**Description**
Client ID and Client Secret	The REST API Client ID and its secret of your PingOne for Enterprise account, see *View or Renew Directory API Credentials* in the *PingOne for Enterprise Administration Guide*.
Advanced Fields	
PingOne URL	The PingOne 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`.

Field	Description
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 Directory. The default value is `100`.

Configuring the RADIUS Username Password Credential Validator

About this task

The RADIUS Username Password Credential Validator verifies credentials using the RADIUS protocol.

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.

> **Tip:** RADIUS server messages are used by the HTML Form Adapter to determine the two-step authentication scenarios and to present a login screen to the end users.

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

Steps

1. Configure one or more RADIUS servers.
   a. Click **Add a new row to 'RADIUS Servers'**.
   b. Enter information into the required fields.

      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, you can 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. This field has no default value.
Authentication Port	The UDP port used to authenticate to the RADIUS server. The default value is `1812`.
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**.

Field	Description
Shared Secret	The password shared between PingFederate and the RADIUS server used to encryptThe attribute identifying the NAS (Network Access Server) originating the request for access. for access. This field has no default value.

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 The password shared between PingFederate and the RADIUS server used to encrypt passwords. `<pf_install>/pingfederate/bin/run.properties` file. Only IPv4 addresses are supported.

c. Click **Update** under **Action**.
d. Repeat these steps to add more RADIUS servers as needed.

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.

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

About this task

The Simple Username Password Credential Validator verifies credentials maintained by PingFederate.

Note: This validator is best used for testing purposes or for an organization with few accounts.

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

Steps

Configure one or more user credentials.

a. Click **Add a new row to 'Users'**.
b. Enter a username, followed by a password (twice).

c. Click **Update** under **Action**.
d. Repeat these steps to add more user credentials as needed.

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.

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. The credential validation process fails when no match is found.

Extending the contract for the credential validator

About this task

In some cases, you might want to extend contracts of the Password Credential Validator 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® 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.

Tip: 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.

Vendor specific RADIUS attributes can be made available by extending the RADIUS attribute dictionary. Copy the vendor-specific attribute dictionaries into the `pingfederate/server/default/conf/radius` directory. The format of the dictionaries must use the *FreeRADIUS* dictionary syntax (freeradius.org/radiusd/man/dictionary.html). Then edit the existing `dictionary` file to include each of them.

Steps

- Optional: On the **Extended Contract** screen, enter an attribute name and click **Add**.

  Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing attribute. Click **Delete** to remove an existing attribute.

Finishing the Password Credential Validator instance configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Configuring Active Directory domains or Kerberos realms

About this task

From the **Manage AD Domains/Kerberos Realms** screen, provide PingFederate with a centralized configuration to authenticate users via the following IdP adapters or token processors:

- **PingFederate integrated Kerberos Adapter** – Using the built-in Kerberos Adapter with a configured Active Directory (AD) Domain allows a PingFederate IdP server to perform SSO to SP applications based on Kerberos tickets.
- **PingFederate integrated Kerberos Token Processor** – The built-in Kerberos Token Processor accepts and validates Kerberos tokens via a configured Kerberos Realm from a web service client.
- **Integrated Windows Authentication (IWA) Integration Kit** (version 3.0 and later) – Using the separately available IWA Adapter with a configured AD Domain allows a PingFederate IdP server to perform SSO to SP applications based on IWA credentials.

Follow these steps to configure an AD domain or Kerberos realm:

Steps

1. Configure the AD environment to integrate with PingFederate (see *Configuring the Active Directory environment* on page 210).
2. Click **Add Domain/Realm** to create an AD domain.

   Important: Do not configure subdomains if the parent domain in the same forest has already been configured (see *Multiple-domain support* on page 210).

   Click the name to edit an existing domain. 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, 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 *How Domain and Forest Trusts Work* (technet.microsoft.com/en-us/library/cc773178(v=ws.10).aspx) from Microsoft.

Note: If you are configuring only one domain, then you also need to configure only one Service Principal Name (see *Configuring the Active Directory environment* on page 210).

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 a separate domain and then map these adapter or token processor instances to your SP connections that authenticate using the integrated Kerberos Adapter, the integrated Kerberos Token Processor, or the (separately available) IWA Adapter.

For information about configuring the PingFederate Integrated Windows Authentication (IWA) adapter for multiple-domain Active Directory trusts, see *https://support.pingidentity.com/s/article/How-to-configure-IWA-with-multiple-Active-Directory-trusts*.

Configuring the Active Directory environment

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 has already been configured (see *Multiple-domain support* on page 210).

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 the password be set with no expiration.
2. Use the Windows utility **`setspn`** to register 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>

   The canonical name of the PingFederate server.

   (For more information on "canonical name", see *https://tools.ietf.org/html/rfc2181#section-10*.)

   <pf-server-account-name>

   The domain account you want to use for Kerberos authentication.

   Note: When executing the **`setspn`** command, `HTTP` must be capitalized and followed by 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 end-users already authenticated must re-authenticate (close the browser or log off and back on) before attempting SSO.

Adding a domain

About this task

Use the **Manage Domain/Realm** screen to configure Active Directory domains or Kerberos realms that PingFederate can use to contact the domain controllers or the Key Distribution Centers (KDCs) for verifying user authentication.

Steps

Enter the required information based on the following table:

Field	Description
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.

Field	Description
Domain Controller/Key Distribution Center Host Names	Specify the host name or IP address of your domain controller or KDC (for example, `dc01-yvr`), and then click **Add**. Repeat this step to add multiple servers.
(optional)	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 anymore. This check box is not selected by default.
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/KDCs is successful, the message `Test Successful` appears. Otherwise, the test returns error messages near the top of the screen. **Tip: Debug Log Output** check box on the **Manage Domain/Realm SettingsFor help resolving connectivity issues, select the** screen, run the test again, and review the debug messages in the PingFederate server log. Note that this test stops at the first successful result when multiple domain controllers or KDCs are specified; therefore, not all servers are necessarily verified. Depending on the network architecture, the engine nodes deployed in a cluster may also establish connections differently. As a result, the engine nodes and the console node may connect to different domain controllers or KDCs.

Managing domain connectivity settings

About this task

Use the **Manage Domain/Realm Settings** screen to change default security and logging settings for all configured Active Directory domains and Kerberos realms.

Steps

Optional: Change the default transport protocol, the debug option, the timeout value, and the number of retry attempts, as needed. For more information of each field, 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 acknowledgement that packets are properly received. **Note:** If you choose this option, ensure that you restart PingFederate after saving the configuration.

Field	Description
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)	Specifies the amount of time (in seconds) PingFederate waits for a network response from a domain controller or KDC. The default is `3` seconds. ⓘ **Note:** This value applies to each attempt PingFederate makes to contact the domain controller or KDC. ⓘ **Note:** The new timeout takes effect only after PingFederate is restarted, after you save the configuration.
AD Domain Controller/ Key Distribution Center Retries	Specifies the number of times PingFederate tries contacting the domain controller or KDC. The default is `3` times.

Managing CAPTCHA settings

About this task

Configure CAPTCHA settings. PingFederate supports invisible reCAPTCHA from Google.

Steps

1. Enter the site key assigned to your account by Google.
2. Enter the associated secret key.

   (There are no default values.)

Managing SMS provider settings

About this task

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 the **Identity Provider# Adapters** screen.
2. Select any HTML Form Adapter instance, in which the selected password reset type is **Text Message**.
3. Click **Manage SMS Provider Settings**.
4. On the **SMS Provider Settings** screen, enter the required information.

Field	Description
Account SID	The account number assigned to your account by Twilio.
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).

Field	Description
From Number	The sender number in the text message notifications.

ⓘ **Tip:** For additional information about each field or Twilio, please refer to *support.twilio.com*.

Results

Once saved, these SMS provider settings apply to all services using text message notifications.

Managing notification publisher instances

About this task

PingFederate delivers messages to administrators and end users based on notification publisher settings. Depending on your use cases, you may create one or more notification publisher instances on the **System#** **Notification Publishers** screen. For example, you may 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

- 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 elect an existing instance to be the default notification publisher instance, click **Set as Default** under **Action**.

Defining a notification publisher instance

About this task

Define a notification publisher instance on the **Type** screen.

Steps

1. Enter an instance name and an instance ID.
2. Select the type of this notification publisher instance from the list.

Configuring a notification publishers instance

About this task

This configuration varies depending on the type of the notification publisher.

Steps

- Refer to subsequent topics for configuration steps.

Configuring an Amazon SNS Notification Publisher instance

About this task

Amazon Simple Notification Service (Amazon SNS) is a messaging service in the Amazon Web Service (AWS) ecosystem. Publishers send their messages to the applicable SNS topics. Subscribers consume those messages as notifications from Amazon via various protocols (such as Lambda and Amazon SQS) by subscribing to the topics that matter to them. 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, please refer to AWS documentation on *Amazon SNS* (docs.aws.amazon.com/sns/latest/dg/welcome.html).

Steps

1. Go to the **System**# **Notification Publishers** screen.
2. Click **Create New Instance** to create a new instance of the **Amazon SNS Notification Publisher**.

   To modify an existing instance, select it by its name under **Instance Name** instead.
3. On the **Instance Configuration** screen, 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 this format: `arn:aws:[service]:[region]:[accountid]:[resourceType/resourcePath]`
Max Payload Size	The maximum payload size in kilobytes. Click **Show Advanced Fields** to reveal this field. Enter a value between 1 and 8192. The default value is `256`.

Results

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. For more information, refer to the subsequent topic.

Event types and variables

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 found 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 that have been configured with an email address) ▪ NOTIFY (represents the **Notify Administrator of Account Change** option on the **Administrative Accounts** screen) ▪ CURRENT_USER_MESSAGE (represents the username of the administrator who initiated the change) **Note:** Unless otherwise noted, the rest of the variables in this **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. The intent of them is 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 instance involved) **Note:** Unless otherwise noted, the rest of the variables in this **HTML Form Adapter instances** section are either self-explanatory or identical to those mentioned here.
PASSWORD_CHANGED	▪ GIVEN_NAME ▪ USERNAME ▪ RECEIVER ▪ ADAPTER_ID ▪ PCV_ID
PASSWORD_RESET	▪ USERNAME ▪ RECEIVER ▪ ADAPTER_ID ▪ PCV_ID ▪ STATUS
PASSWORD_RESET_FAILED	▪ USERNAME ▪ RECEIVER ▪ ADAPTER_ID ▪ PCV_ID
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 username of the end user who should receive an email ownership verification request) ▪ RECEIVER (represents the email address to which the email ownership verification request should be 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

About this task

You can set up an instance of the SMTP Notification Publisher for PingFederate uses to notify administrators and end users about various events. As needed, you may configure multiple instances, each with different settings.

Steps

1. Go to the **System**# **Notification Publishers** screen.
2. Click **Create New Instance** to create a new instance of the **SMTP Notification Publisher**.

   To modify an existing instance, select it by its name under **Instance Name** instead.
3. On the **Instance Configuration** screen, 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 must be in valid format but need not be set up on your system.
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) to establish an unencrypted connection to the email server at the SMTP port.
SMTPS Port	The secure SMTP port on your email server. Inapplicable unless **SSL/TLS** is the chosen encryption method. The default value is `465`.

Field	**Description**
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. Inapplicable unless unless either **SSL/TLS** or **STARTTLS** is the chosen encryption method. This check box is selected by default.
Username	Authorized email account.
Password	Password for the authorized email account.
Test Address	Enter an email address the 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`.
Retry Attempts	The number of times the PingFederate tries again after encountering an error. The default value is `2`.
Retry Delay (Minutes)	The amount of time in minutes that PingFederate waits before trying to send an email message again. The default value is `2`.
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 may contain sensitive information, to the server log. Consider enabling debug messages solely for troubleshooting purpose and disabling this option when debug messages are no longer required.

Results

As needed, you can modify email-notification template files to suit the particular branding requirements. For more information, see *Customizable email notifications* on page 268.

Finalizing actions for a notification publisher instance

Steps

On the **Actions** screen (if shown), follow the on-screen instructions provided by the developer to complete any required tasks.

Depending on the data store implementation, configuration requirements vary. If no action is required, this screen is not shown.

Reviewing a notification publisher instance configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.
- To discard your changes, click **Cancel**.

Results

Once set up and saved, you may select this notification publisher instance in any component that is capable of triggering or handling events.

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. This file is 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, refer to the file itself.

Tip: The clustering configuration options are also maintained in the `run.properties` file. For more information, see *Configure cluster protocol properties* on page 913.

Property	Description
pf.admin.https.port	Defines the port on which the PingFederate administrative console runs. The default value is `9999`.
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, used to make separate instances identifiable.
pf.console.session.timeout	Defines the length of time in minutes until an inactive administrative console times out. The minimum setting is 1 minute; 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 log on to PingFederate using credentials managed internally by PingFederate or externally by other systems.
pf.admin.api.authentication	Defines the authentication method of the PingFederate administrative API.

Property	Description
ldap.properties.file	When 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.
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`.
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 (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 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 an SNMP agent and 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 configure 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; for example: `*.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 the **System**# **Protocol Settings**# **Federation Info** screen.) 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 the **System**# **Virtual Host Names** screen.) 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. If you want 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, the runtime server is accessible 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.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), keys and certificates on are stored on the HSM when applicable. The HSM hybrid mode allows an organization 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 *Transition to an HSM* on page 313.
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 (see *Customize the heartbeat message* on page 290).
org.apache.xml.security.ignoreLineBreaks	When set to `true` (the default), PingFederate does not insert line breaks in XML digital signatures.

ⓘ **Note:** Additional configuration of the listener ports (including adding new listeners) is available via 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.

   Consider creating a backup copy of the file.
2. Modify the applicable properties.
3. Restart PingFederate.

Results

ⓘ **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; it is also *not* part of the **Replicate Configuration** process. PingFederate must be restarted if running.

PingFederate log files

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 purpose.

`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 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 server to use the same logging configuration, manually synchronize the `log4j2.xml` file across multiple PingFederate server.

Log levels and verbosity

Log messages are categorized into six log levels:

1. `FATAL`
2. `ERROR`
3. `WARN`
4. `INFO`
5. `DEBUG`
6. `TRACE`

Starting with version 8.2, PingFederate only records messages that are tagged with log level `INFO`, `WARN`, `ERROR`, and `FATAL` to the server log (and the provisioner log). Messages that are tagged `DEBUG` (or `TRACE`) are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purpose, you may adjust the log level to `DEBUG` in the `log4j2.xml` file and optionally re-enable console logging.

Important: When debug messages and console logging are no longer required, ensure they are turned off.

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 *Enable debug messages and console logging* on page 879.

Changes, such as adding a `Logger` or adjusting log levels, are activated within half a minute. No restart of PingFederate is required.

Fields (and attributes)

Some logs, such as the audit log and the administrative API log, can be customized to log additional (or less) information by modifying their `pattern` elements. Available fields are documented inline in the `log4j2.xml` file.

ⓘ **Tip:** PingFederate can be configured to log user attributes (if they are present) in the audit log, transaction log, and server log. When privacy is required 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 useful for troubleshooting and support purposes to aggregate and analyze log entries tied to the same original request. The tracking ID (`%X{trackingid}`) may also be added to the configuration for the transaction log, or be 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 CEF format. Furthermore, the audit log may also be written in a format that can be 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 may 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. As needed, administrators may adjust the file size and the number of files to be retained in the `log4j2.xml` configuration file.

For more information about Log4j 2, please refer to the *Log4j 2 open-source project* (logging.apache.org/log4j/2.x/manual/index.html).

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)

These configuration files are located in the `<pf_install>/pingfederate/etc` directory and independently managed 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. While the events themselves are not configurable, Log4j 2 configuration settings (in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file) may be adjusted to deliver the desired level of detail surrounding each event.

Events logged by PingFederate includes (but not limited to):

- Login 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
- IdP/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

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 the corresponding **Save** button in the administrative console is clicked. 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 role(s) assigned to the administrator at the time the event occurred.
- The type of event that occurred.
- Basic information about the event.

Each of the above fields is separated by a vertical pipe (`|`) for easier parsing.

Detailed event logging

PingFederate can also be configured to log additional event information to a separate log file. When detailed event logging is enabled, besides writing basic information to `<pf_install>/pingfederate/log/admin.log`, PingFederate logs detailed information about each event to `admin-event-detail.log` (in the same `log` directory). Events between `admin.log` and `admin-event-detail.log` are linked by a unique event ID. Each entry in `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.

Note: Not all events have detailed information; for example, login attempts are only logged to `admin.log`.

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). 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 via the administrative API. This information is recorded in the `<pf_install>/pingfederate/log/admin-api.log` file. While the events themselves are not configurable, Log4j 2 configuration settings (in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file) may 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

Each of the above fields is separated by a vertical pipe (`|`) for ease of parsing.

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 themselves are not configurable, Log4j 2 configuration settings (in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file) may 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

Each of the above fields is separated by a vertical pipe (`|`) for ease of parsing.

Runtime transaction logging

About this task

PingFederate provides for flexible, scalable logging of all federated-identity transactions (inbound and outbound messages). Administrators may configure transaction logging to any of the four modes on a per-connection basis or override the logging mode for all SP connection, 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**
None	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 SSO transaction)
Enhanced	Includes everything logged at the **Standard level** plus: ▪ SAML_SUBJECT* ▪ Binding ▪ Relay state (if available) ▪ Signature policy ▪ Signature status ▪ HTTP request parameters (outbound messages only) * Only when available in a SAML assertion, a single-logout 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:
  a. Select the applicable connection from the **Identity Provider** or **Service Provider** screen.
  b. Click **General Info** and then select the one of the four logging modes.
- To override transaction logging mode for all SP (or IdP) connections:
  a. On the **Identity Provider** (or **Service Provider**) screen, click **Manage All** under **SP Connections** (or **IdP Connections**).
  b. Turn on the **Logging Mode Override** setting and select a logging mode for all SP (or IdP) connections.

Security audit logging

PingFederate records a subset of transaction log information with additional details at runtime, intended to facilitate security auditing and regulatory compliance. Activities from SSO, SLO, OAuth, WS-Trust STS, and SCIM inbound provisioning transactions are recorded in the security audit log, the `audit.log` file, located in the `<pf_install>/pingfederate/log` directory. Security audit log information may be output 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.
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 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.
subject	The subject of the transaction or authentication attempt.
ip	The incoming IP address.
app	The target 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.

Field	Description
adapterid	The ID of an adapter instance. Consider adding the authenticationsourceid and targetsessionid fields to record additional information about the request.
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.
responsetime	The time elapsed (in milliseconds) from when a final request for a transaction is received to when the audit message is written. This value serves as an approximation of total transaction processing time and may 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.
attrackingid	The tracking ID for OAuth access token. It could be used 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.
granttype	The OAuth grant type.

Field	Description
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.
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.
localuserid	The local ID used for the transaction (when account linking is enabled at the SP).
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.

Field	Description
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.
responseid	The ID of a SAML response.
requeststarttime	The start time of the request in milliseconds since midnight, January 1, 1970 UTC.
sessiongroupid	The internal ID for a group of persistent authentication sessions associated with a single browser instance via the PF.PERSISTENT cookie. It is only set if the request has triggered a session lookup.
sri	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.
stspluginid	The ID for the token processor or token generator instance. Applicable only to WS-Trust STS transactions.
targetsessionid	An array of one or more SP adapters or SP connections (or both) invoked in an authentication or logout flow.

Field	Description
trackedparameter{*anHttpRequestParameter*}	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 **Authentication Policies# Tracked HTTP Parameters** screen. For more information about tracked parameters, see *Policies* on page 339. 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.
validatorid	The ID of the Password Credential Validator instance (for the successful attempts).
virtualserverid	The virtual server ID of a request (if applicable).

Outbound provisioning audit logging

The PingFederate `provisioner-audit.log` file records outbound provisioning transactions, intended to facilitate security auditing. The log file is located in the `<pf_install>/pingfederate/log` directory. Outbound provisioning audit log information may be output to different formats, including database and Splunk.

The following table describes all recorded elements. As needed, elements are configurable 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.

Server logging

When PingFederate is configured to log `DEBUG` messages (for troubleshooting purpose), it records all runtime and administrative events, including status and error messages that can be used for troubleshooting, in the `<pf_install>/pingfederate/log/server.log` file. Server log information may be output to a database server.

Note: `DEBUG` messages are turned off by default. For troubleshooting purpose, you may re-enable it by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

The following table describes the recorded elements. As needed, elements are configurable 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.

Server log filter

PingFederate provides a utility **`logfilter`** that administrators can use to filter server logs. The utility 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*> may be the value of any of the following transaction tags in the target server log(s): ▪ 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. Administrators can control settings for this log, as needed, in the file `logfilter.log4j2.xml`, located in the `<pf_install>/pingfederate/bin` directory.

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 may 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

These scripts are located in the `<pf_install>/pingfederate/server/default/conf/log4j/sql-scripts` directory.

Note: PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see . To obtain your database driver JAR file, contact your database vendor. Database driver file should be installed to the `<pf_install>/pingfederate/server/default/lib` directory. You must restart the server after installing the driver.

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. Under the **Preserve messages in a local file** section, for each log that you want to enable database logging, uncomment the preset `JDBC` appender configuration based on the choice of your database server.

   Audit log

   - `SecurityAuditToMySQLDB` (for Oracle MySQL)
   - `SecurityAuditToOracleDB` (for Oracle Database)
   - `SecurityAuditToPostgreSQLDB` (for PostgreSQL)
   - `SecurityAuditToSQLServerDB` (for Microsoft SQL Server)

Provisioner audit log

- `OutboundProvisionerEventToMySQLDB` (for Oracle MySQL)
- `OutboundProvisionerEventToOracleDB` (for Oracle Database)
- `OutboundProvisionerEventToPostgreSQLDB` (for PostgreSQL)
- `OutboundProvisionerEventToSQLServerDB` (for Microsoft SQL Server)

Provisioner log

- `ProvisionerLogToMySQLDB` (for Oracle MySQL)
- `ProvisionerLogToOracleDB` (for Oracle Database)
- `ProvisionerLogToPostgreSQLDB` (for PostgreSQL)
- `ProvisionerLogToSQLServerDB` (for Microsoft SQL Server)

Server log

- `ServerLogToMySQLDB` (for Oracle MySQL)
- `ServerLogToOracleDB` (for Oracle Database)
- `ServerLogToPostgreSQLDB` (for PostgreSQL)
- `ServerLogToSQLServerDB` (for Microsoft SQL Server)

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 reasons. Both appenders must also be enabled (uncommented).

Tip: Review inline comments and notes in the `log4j2.xml` file for more information about each appender.

3. Replace placeholder parameter values in `log4j2.db.properties` in the same `conf` directory for the applicable 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:

   - `org.sourceid.websso.profiles.sp.SpAuditLogger` (Browser SSO SP and adapter-to-adapter)
   - `org.sourceid.websso.profiles.idp.IdpAuditLogger` (Browser SSO IdP and adapter-to-adapter)
   - `org.sourceid.websso.profiles.idp.AsAuditLogger` (OAuth authorization server)
   - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger` (Dynamic Client Registration)

- `org.sourceid.wstrust.log.STSAuditLogger` (WS-Trust STS, IdP and/or SP)

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

The Common Event Format (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, using syslog, when available, is recommended.

Note: PingFederate is tested with ArcSight for interoperability using the default elements defined in `log4j2.xml`. Any additions to these elements may render your CEF logging incompatible with ArcSight.

Writing audit log in CEF

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 (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:
   - `org.sourceid.websso.profiles.sp.SpAuditLogger` (Browser SSO SP and adapter-to-adapter)
   - `org.sourceid.websso.profiles.idp.IdpAuditLogger` (Browser SSO IdP and adapter-to-adapter)
   - `org.sourceid.websso.profiles.idp.AsAuditLogger` (OAuth authorization server)
   - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger` (Dynamic Client Registration)
   - `org.sourceid.wstrust.log.STSAuditLogger` (WS-Trust STS, IdP and/or SP)

   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

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 log for Splunk

Ping Identity provides a custom Splunk App for PingFederate to process audit logs generated by a PingFederate deployment.

About this task

Splunk is enterprise software that allows for monitoring, reporting, and analyzing consolidated log files. Splunk captures and indexes real-time data into a single searchable repository from which 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 IdP and SP reports per connection
- Trend reports such as weekly and monthly usage reports, and trend analysis

The application uses a specially formatted version of the audit log (`splunk-audit.log`), which is written to the PingFederate log directory when the setup steps described below are followed.

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. Download the free application from *splunkbase.splunk.com*. (Search for PingFederate.)

Steps

1. Set up your Splunk server.
   a. If you have not done so, download and install Splunk.
   b. Enable a receiver to *listen* for data from the PingFedrate server.

      For more information, please refer to *Splunk documentation* (docs.splunk.com/Documentation/Forwarder/7.1.2/Forwarder/HowtoforwarddatatoSplunkEnterprise).
   c. Install Splunk App for PingFederate.
2. Configure PingFederate to write audit log messages to the `<pf_install>/pingfederate/log/splunk-audit.log` file.
   a. Edit `<pf_install>/pingfederate/server/default/conf/log4j2.xml`.
   b. Look for the following `Logger` elements located under the **Loggers** section:

      - `org.sourceid.websso.profiles.sp.SpAuditLogger` (Browser SSO SP and adapter-to-adapter)
      - `org.sourceid.websso.profiles.idp.IdpAuditLogger` (Browser SSO IdP and adapter-to-adapter)
      - `org.sourceid.websso.profiles.idp.AsAuditLogger` (OAuth authorization server)
      - `org.sourceid.websso.profiles.idp.ClientRegistrationAuditLogger` (Dynamic Client Registration)
      - `org.sourceid.wstrust.log.STSAuditLogger` (WS-Trust STS, IdP and/or SP)
   c. Uncomment the `SecurityAudit2Splunk` `RollingFile` appender reference (`<appender-ref ref="SecurityAudit2Splunk"/>`) from the one or more of the `Logger` elements.

      For example, the default logger for an IdP audit log reads:

```
<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.

d. 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.

a. Download the Splunk Universal Forwarder from *Splunk* (www.splunk.com/en_us/download/universal-forwarder.html) and install it on the PingFederate server.

b. Configure the Splunk Universal Forwarder to monitor the `splunk-audit.log` file and forward the data to the receiver configured in *step 1b*.

For detailed installation and configuration instructions, please refer to *Splunk documentation* (docs.splunk.com/Documentation/Forwarder/7.1.2/Forwarder/HowtoforwarddatatoSplunkEnterprise).

Alternative console authentication

As an alternative to using PingFederate's own internal data store for authentication to the administrative console, you can configure PingFederate to use either your network's LDAP user-data store, the RADIUS protocol, or client certificates. You can configure any of these alternatives at any time.

Note that 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 the **System# Administrative Accounts** screen, 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	Admin	Configure partner connections and most system settings (except the management of local accounts and the handling of local keys and certificates).
Admin	Crypto Admin	Manage local keys and certificates.
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Auditor	*Not applicable*	View-only permissions for all administrative functions. When the **Auditor** role is assigned, no other administrative roles may be set.

ⓘ **Note:** All three administrative roles are required to access and make changes through the following services:

- The `/ConfigArchive` administrative API endpoint
- The **System**# **Configuration Archive** screen in the administrative console
- The **Connection Management** configuration item on the **Security**# **Service Authentication** screen

Enabling LDAP authentication

You can enable LDAP authentication by using 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 login attempts. Responsibility for preventing access is instead delegated to the LDAP server and enforced according to its password lockout settings.

Steps

1. In the `<pf_insall>/pingfederate/bin/run.properties` file, change the value of the pf.console.authentication property as shown below:

   ```
   pf.console.authentication=LDAP
   ```
2. In the `<pf_install>/pingfederate/bin/ldap.properties` file, change property values as needed for your network configuration.

   See the comments in the file for instructions and additional information.

   Note that the roles configured in the properties file apply to both the administrative console and the administrative API.

   ⓘ **Important:** Be sure 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.

   ⓘ **Tip:** You can also use this configuration file in conjunction with RADIUS authentication to determine permissions dynamically via an LDAP connection.
3. Start or restart PingFederate.

Enabling RADIUS authentication

You can enable RADIUS authentication by using 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 (by appending to the password a one-time passcode obtained from an authenticator, for instance) and two-step authentication (through a challenge-response process, for example).

ⓘ **Note:** When RADIUS authentication is configured, PingFederate does not lock out administrative users based upon the number of failed logon attempts. Responsibility for preventing access is instead 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_insall>/pingfederate/bin/run.properties` file, change the value of the pf.console.authentication property as shown below:

   ```
   pf.console.authentication=RADIUS
   ```
2. In the `<pf_install>/pingfederate/bin/radius.properties` file, change property values as needed for your network configuration.

   See the comments in the file for instructions and additional information.

   Note that 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 (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 (also in the `bin` directory) to map LDAP group-based permissions to PingFederate roles.
3. Start or restart PingFederate.

Multifactor console authentication using PingID

The PingFederate administrative console supports authentication via the RADIUS protocol, which provides a common approach for implementing strong authentication in a client-server configuration. *PingID®* is a cloud service that enables multifactor authentication using a mobile application.

By combining these two capabilities, PingID can be configured to provide multifactor authentication (MFA) to protect access to the PingFederate administration 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*.

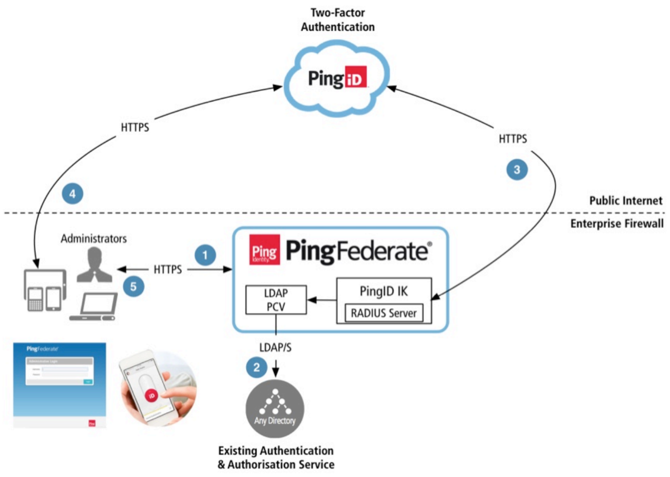
Requirements

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

When PingID® is the second authentication factor for the PingFederate administrative console, the administrators must authenticate successfully against the first factor (for example, a directory server) and subsequently respond to the request for authentication from the PingID app on their mobile devices.



MY TITLE Multifactor Console Authentication using PingID

Processing steps

1. An Administrator opens a browser and accesses the PingFederate administration console.
   a. The administrative console displays the login page.
   b. 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 data store.

   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 screen. 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 an SMS or voice callback message, depending on the configuration for that user account.
   a. The administrator responds to the request for authentication from PingID.
   b. 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 opens its menu.

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. In addition, you must configure the account to allow the PingID Password Credential Validator (PCV) to use the PingID service.

Steps

1. Register a PingID account.
   a. Contact sales@pingidentity.com for a registration key.
   b. Register at *PingOne® admin portal* .
   c. 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.
   d. Open the confirmation email and follow the instructions within it to activate your PingOne for Enterprise account.
2. Enable PingID client integration.
   a. Sign on to the *PingOne admin portal* .
   b. Select **Setup**# **PingID**# **Client Integration** screen.
   c. Under **Integrate with PingFederate and other Clients**, click **Download** and save the `pingid.properties` file for a subsequent task.
   d. Sign off.

Creating an LDAP Username Password Credential Validator instance

About this task

The administrators must authenticate successfully against the first factor; for example, 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 the **System**# **Password Credential Validators** screen, and then click **Create New Instance**.
2. On the **Type** screen, select **LDAP Username Password Credential Validator** from the list and provide a name and an ID for it.
3. On the **Instance Configuration** screen, select the LDAP data store and enter information into the required fields.

   For more information about each field, refer to the following table:

Field	Description
LDAP Datastore (Required)	The LDAP data store configured in PingFederate. If you have not yet 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 from which the search begins. This field has no default value.

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 may 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 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 screen. This check box is selected by default.

4. On the **Extended Contract** screen, click **Next** to skip to the **Summary** screen.
5. On the **Summary** screen, review the configuration, modify as needed, and then save the configuration.

Configuring a PingID Password Credential Validator instance

About this task

In this step, create and configure an instance of the PingID® Password Credential Validator (PCV).

Steps

1. Go to the **System**# **Password Credential Validators** screen, and then click **Create New Instance**.
2. On the **Type** screen, select **PingID PCV (with integrated RADIUS server)** from the list and provide a name and an ID for it.
3. On the **Instance Configuration** screen, configure the required fields as follows.
   a. 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, and then 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.
   b. Click **Add a new row to 'Delegate PCV's'**, select the previously created LDAP Username Credential Validator instance, and then click **Update**.
   c. Open the previously downloaded `pingid.properties` file in a text editor, copy its content, and then close the file.
   d. Scroll down to the end of the **Instance Configuration** screen, and then paste the content from the `pingid.properties` file into the **PingID Properties File** text field.
   e. Review the rest of the default settings. Modify as needed to meet your requirements.

4. On the **Extended Contract** screen, click **Next** to skip to the **Summary** screen.
5. On the **Summary** screen, review the configuration, modify as needed, and then save the configuration.

Configuring PingFederate to use RADIUS authentication

About this task

In this multifactor 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. Follow these steps to enable RADIUS 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`; for example:

   ```
   pf.console.authentication=RADIUS
   ```
2. Obfuscate the **Client Shared Secret** value using a PingFederate command-line tool:

   `<pf_install>\pingfederate\bin\obfuscate.bat` *`clientSharedSecret`* on Windows

   `<pf_install>/pingfederate/bin/obfuscate.sh` *`clientSharedSecret`* on Linux

   The output should be a long line of text. Copy the output for the next step.
3. Open the `<pf_install>/pingfederate/bin/radius.properties` file in a text editor and modify as follows:

   ```
   host=localhost
   shared.secret=obfuscatedClientSharedSecret
   timeout=10000
   ```

   Tip: 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 (`10000` ms) should be sufficient for PingID.

   In addition, assign one or more RADIUS users or designated RADIUS groups, or both, 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.
4. Save your changes, and then restart PingFederate.

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 following 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 reachable 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 the screen to register a device.
5. Respond to the request for authentication from PingID and the PingFederate administrative console should display its menu.

Results

ⓘ **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

To enable client-certificate authentication, PingFederate administrative users must have imported into their web browsers an X.509 key and certificate suitable for user authentication. In addition, the corresponding root CA certificate(s) must be contained in the Java runtime or the PingFederate trusted store.

About this task

Other setup steps, including designating user permissions, must be completed by using configuration files located in the `<pf_install>/pingfederate/bin` directory.

Note that the roles configured in the properties file apply to both the administrative console and the administrative API.

Steps

1. If not already done, import the necessary client key and certificate into the web browser used to access PingFederate.
2. Log on normally (using username and password) to the PingFederate console as a user with permissions that include the **Crypto Admin** role.
3. 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).

   You can import a certificate to PingFederate in the **Security# Trusted CAs** screen.

   ⓘ **Tip:** You may wish to click the Serial number and copy the Issuer DN to use in a couple steps later.
4. In the `pingfederate/bin/run.properties` file, change the value of the pf.console.authentication property as shown below:

   ```
   pf.console.authentication=cert
   ```
5. 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`.

   If you copied the Issuer DN a couple steps earlier, paste this value.

   See the comments in the file for instructions and additional information.

   Note that the roles configured in the properties file apply to both the administrative console and the administrative API.
6. Repeat the previous step for any additional CAs as needed.

7. 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.

8. Repeat the previous step for all users as needed.

   > ⓘ **Note:** Other settings in the properties file are used to display the user's ID (the Subject DN) in abbreviated form in the administrative console.

9. 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 affected adversely by subsequent changes in supporting components.

About this task

This type of error checking occurs whenever you access certain supporting components, such as the **Adapters** screen, the **Token Processors** screen, the **Token Generators** screen, the **Password Credential Validators** screen, and the **Identity Store Provisioners** screen.

As the number of connections and supporting components increases, so does the validation time. If you experience noticeable delays in accessing 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. While you are free to make configuration changes without being prompted to fix all dependency errors immediately, keep in mind that the configuration changes could potentially introduce service disruption.
>
> For example, if you remove an attribute that has been configured as a source-attribute in an attribute fulfillment configuration in a connection (or connections), users will not be able to complete SSO requests until you reconfigure such connection (or connections). When you access the **Connections** screen, 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 that requires your attention.

This setting does not affect the validation of a connection as it is being configured or modified. Moreover, individual connections are always validated automatically when you access them on the **Connections** screen, regardless of the configuration of this setting.

You manage this setting on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

Steps

- To 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 both the **Identity Provider**# **SP Connections** screen and the **Service Provider**# **IdP Connections** screen.

Automating configuration migration

PingFederate provides a configuration-migration tool that can be used for scripting the transfer of administrative-console configurations and configuration property files from one PingFederate server to another; for example, from a test environment to production. The tool may also be used to manage certificates for the target server.

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 (see *Connection Management Services*).

Important: The Connection Management Service must be activated for both the source and target servers before the **`configcopy`** tool can be used (see *Configure service authentication* on page 321).

CAUTION: For security reasons, the Connection Management Service should be disabled 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

To do this, you copy the key (or keys, if you have more than one) 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 done 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.

   Ensure you copy the entire key JSON message. For example, you would copy the text as shown in bold below:

   `{"keys":[`**{"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:

   ```
   {"keys":
   [{"kty":"oct","kid":"wER9zEpaPe","k":"i0HQr9JmsqjAX4o_BQU1qGJzoLQI-
   nmwp8u3GyHzTB8"}]}
   ```

   Insert the comma and the source key as follows:

   ```
   {"keys":
   [{"kty":"oct","kid":"wER9zEpaPe","k":"i0HQr9JmsqjAX4o_BQU1qGJzoLQI-
   nmwp8u3GyHzTB8"},
   ```

```
{"kty":"oct","kid":"j0PUEdAb95","k":"AGi8Lg_ewdl-
_30Cx83kDMQE9oNlhgJSa_Pc4I8JTU8"}]}
```

Note that 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

For migrating data configured with the source server's administrative console, this tool performs these overall processing steps:

1. Retrieves specified connection, other configuration data (XML), or both, 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, we recommend using 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 data-store 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 data-store 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 may have been manually customized for the source configuration and need to be migrated. The file-copy command may also be used to copy the PingFederate internal, HSQLDB database when needed.

Managing Certificates

Administrators may use the **`configcopy`** tool to perform the following certificate-management tasks on the target PingFederate server:

- List source trusted 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/or target PingFederate servers, and to modify configuration settings as needed for the target environment.

About this task

Property-file templates are available for each command option. The template files are located in the `<pf_install>/pingfederate/bin/configcopy_templates` directory.

Note: Refer to the `README.txt` file in the `configcopy_templates` directory for a list of all commands and summary information. See the template files themselves for parameters associated with each command (or with use cases), as well as lists of Override Properties (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>;...
```

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 that the file separators are platform specific, corresponding to the syntax used for system-level path separators.

Alternatively (or in addition), you can specify any property values via 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 may be set in the properties file for both the source and target web services (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 always must 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 attempt 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.

In addition, to avoid errors and prevent unstable target configurations due to missing components or faulty cross-component references (for example, invalid ID references from connection configurations to data-store configurations), be sure to adhere closely to the instructions provided in the following procedure.

Steps

1. Ensure access to the Connection Management Service is enabled for both the source and target PingFederate servers (see *Configure service authentication* on page 321).
2. Determine which component configurations need to be copied, including plug-ins.

   For example, connection configurations always reference either adapter or token-translator configurations (or both) and may reference data-store configurations. These are all separate configurations, and must be copied separately (unless they already exist at the target) in conjunction with copying connection configurations.

   Server Settings, unless pre-configured at the target, also need to be copied over separately.

   Provisioning settings may be copied separately as needed to update target connections.
3. Determine whether any configuration property files or other supporting files need to be copied.
4. Ensure necessary plug-in 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`
5. On the target server, ensure that signing certificates (or certificates used for XML decryption) are already in place (see *Certificate and key management* on page 292).

   Private keys are not copied from server to server (public certificates may be copied); however, you may use **configcopy** to upload keys/certificates to the target server.

   Make note of identifying information about the target keys so you can reference the certificates in connection-copy properties.
6. If you have not yet installed your organization's (CA-issued) SSL server certificate on both the target and source servers, either do so—you can use a **configcopy** command for this—or use one of the following work-arounds to ensure that **configcopy** can contact both servers:

   Either:

   - (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.

     Or:
   - Install the Issuer certificate for the PingFederate SSL certificate into the trust store for the Java runtime under which **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 under which **configcopy** runs.

7. Create properties files for the necessary commands and associated command-parameter values needed to copy the required configurations and any additional files.

   Refer to the REAME.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 may also use command-line parameters (see information earlier in this section).

8. If you are copying connections, override ID properties referencing adapter, data stores or other plug-in configurations, as needed (see *step 2*).

   > **Important:** Ensure that the plug-in 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.

9. Create a script or run a command (or command series) that executes **`configcopy`** for each of the prepared properties files.

   See the discussion above 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 IdP sites. Administrators can use this tool 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 may get 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 internal data store PingFederate uses to maintain provisioning synchronization between the LDAP user store and the target service.

Note that the tool creates its own log file, `provmgr.log`, located in the directory `<pf_install>/pingfederate/log`. You can control settings for this log, as needed, in the file `provmgr.log4j2.xml`, located in the `<pf_install>/pingfederate/bin` directory.

The following table describes the available global and channel-specific command arguments.

Command argument	Description
Global options	
--help	Describes the available options. The help is also displayed if the command is run 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 (for example, `5000/12` 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).

Command argument	**Description**
--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 may 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:** This option should seldom be needed if users are deactivated rather than deleted. If it is needed, you may wish to schedule it when other network activity is low.
--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 option should be needed rarely and may consume considerable network resources, depending on the number of users. If it is needed, you may wish 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. Note, however, that users whose recorded timestamp is unchanged are *not* updated.
--reset-all	Equivalent to using all three of the arguments above. **Important:** This option should be needed rarely if ever and may consume considerable network resources, depending on the number of users. If it is needed, you may wish to schedule it when other network activity is low.
--show-dirty-records	Lists all users or groups that have not been provisioned or updated at the service-provider site.
--show-dirty-group-records	List groups that have not been provisioned or updated at the service-provider site.

Command argument	Description
--show-dirty-user-records	List all users that have not been provisioned or updated at the service-provider 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 (fields waiting to be pushed to the service provider); 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).
	Examples:
	`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.
--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 service provider when the synchronization frequency interval (defined on the **System#** **Protocol Settings#** **Outbound Provisioning** screen) has expired. **Important:** These options should be needed rarely if ever. If needed, you may wish 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 screens

PingFederate supplies HTML templates to provide information to the end users or to request user input when processing their requests. These template pages are located in the `<pf_install>/pingfederate/server/default/conf/template` directory. They utilize the Velocity template engine, an open-source Apache project. For 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. Cascading style sheets 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.

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
`$escape`	A utility class that can be used to escape String variables inserted into the template; for example, `$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; for instance, `window.location.replace("$escape.forJavaScript($wreply)")` as seen in the `sourceid-wsfed-idp-signout-cleanup-template.html` template file. **Important:** We recommend using 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 (for example, the User-Agent HTTP header).
`$HttpServletResponse`	A Java object instance of `javax.servlet.http.HttpServletResponse`. Used to modify the response in the template (for example, setting additional browser cookies).
`$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, we recommend using 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.

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 an SSO error occurs during 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 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 **Identity Provider** or **Service Provider** menu.

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 may also localized these messages using the PingFederate localization framework.

Note: If you have a clustered PingFederate environment, copy the customized (and localized) templates to each node.

IdP user-facing pages

HTML Form Adapter

Page title and *template file name*	Purpose	Type	Action
Sign On or **Choose an Account** *identifier.first.template.html*	Displays to prompt 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 that has been configured to support authentication via third-party identity providers, those identity providers are displayed on the sign-on page as well.	Normal	User input required
Change Password *html.form.change.password.template.html*	Displayed when a user attempts to change their password via the HTML Form Adapter.	Normal	User input required
Change Password *html.form.message.template.html*	Displayed when a user has successfully changed their password.	Normal	User input required

Page title and *template file name*	**Purpose**	**Type**	**Action**
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.	Normal	User input required
Password Management System Message *html.form.message.template.html*	Displayed when a user is being redirected to a password management system to change their password.	Normal	User input required
Account Recovery *forgot-password.html*	Displayed when a user attempts to reset their password via the HTML Form Adapter. If the user has entered a username in the sign-on form, the username is carried over to this form; otherwise, the user must enter their username to being 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 has successfully reset 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 has successfully unlocked 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. This template can be used, for example, to create a RADIUS challenge form when using the RADIUS Username/Password Credential Validator.	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 **Identity Provider**# **Default URL** screen.	Normal	User input required
Sign Off *idp.logout.success.page.template.html*	Displayed when a user has successfully signed 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*	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 has been configured to support self-service registration.	Normal	User input required
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 has been configured to support self-service profile management.	Normal	User input required

Page title and *template file name*	Purpose	Type	Action
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 has been 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 has been configured to offer users the opportunity to verify the ownership of the email address associated with the accounts.	Normal	None
Email Verification Error *local.identity.email.verification.error.html*	Displays that the user has 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 has been 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.

Page title and *template file name*	**Purpose**	**Type**	**Action**
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

Page title and *template file name*	Purpose	Type	Action
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. Note that 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 OpenID Connect client IdP sign-out processing page. Note that 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 OpenID Connect 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

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.emplate.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 SSO, but the 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 **Service Provider**# **Default URLs** screen.	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 an 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

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. Note that 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
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 title and *template file name*	Purpose	Type	Action
Page Expired *state.not.found.error.page.template.html*	Displayed when simultaneous SSO requests (from multiple tabs using the same PF 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 AS provides five screens that are presented to end users (resource owners) during certain OAuth transactions. These screens can be customized and branded 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 (resource owners) to revoke persistent access grants.	Normal	User input required
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 may prompt once at first or repeatedly depending on the **Reuse Existing Persistent Access Grants for Grant Types** setting in the **OAuth Server**# **Authorization Server Settings** screen. 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 client configuration screen. (When an external consent user interface is used, PingFederate does not make use of this template file.)	Normal	User input required

Page title and *template file name*	**Purpose**	**Message type**	**Action**
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
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 notification publisher settings. Each component that is capable of triggering or handling events may use a different notification publisher instance to deliver its messages. For example, you may 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. As needed, you can modify these template files in a text editor to suit the particular branding requirements.

Note: If you have a clustered PingFederate environment, copy the customized templates to each node.

Tip: You may 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* on page 214 and *Event types and variables* on page 215 for more information about this messaging model and the handling of notification messages.

Local administrative account management events

Notification for account management events is configurable on the **System# Administrative Accounts** screen.

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.

Certificate events

If PingFederate is configured to generate notification for certificate expiration events, for any self-signed certificates that the administrators have enabled automatic certificate rotation on the **Security**# **Signing & Decryption Keys & Certificates** screen, PingFederate also generates email notification when a new pending certificate is created and when it is activated. The notification is sent to the email address that has been configured for the certificate events on the **Runtime Notifications** screen.

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** screen. 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 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 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.

Email subject and *template file name*	Event	Action
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

PingFederate supports automatic reloading of SAML metadata, which streamlines the maintenance of SAML connections. When notification for SAML metadata update events is enabled on the **Runtime Notifications** screen, PingFederate sends the notifications to the email address that has been configured for the event on the same screen.

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

When notification for licensing events is enabled, PingFederate sends license expiry information to the recipient configured for the event on the **Runtime Notifications** screen.

Tip: To check the details of your license, sign on to the administrative console, point to the user icon in the upper-right corner of the administrative console, and click **About** the list. The license summary is displayed in a pop-up browser window.

Important: If an expiration date is specified, the license expires at the beginning of the 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 an HTML Form Adapter instance is configured 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 (but **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.

Email subject and *template file name*	Event	Action
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 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.

Customizable text message

If you have configured self-service password reset (and optionally self-service account unlock) to use the **Text Message** option, you may customize (and localized) the text message for a unique experience. The default message is stored as a property in the `pingfederate-sms-messages.properties` file, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

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 three types of end-user messages: on-screen messages, email messages, and text messages (SMS).

About this task

The English contents for each message type are located 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`

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 browser settings.

   For example, to localize the user-facing screen 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 (for example 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 instance, to localize the text messages in Canadian French, renamed the translated copy to `pingfederate-sms-messages_fr_CA.properties`.

   Note: The capitalization and underscore usage may 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.

Results

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. Refer to 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 `locale-options.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

Be sure to return the value to `true` when testing is complete. Note that restarting the server is required for changes to configuration files.

Locale overrides by cookies

For convenience, an administrator or web developer might want to provide end-users a means of overriding browser language preferences temporarily by setting cookies; for example, by 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 via a cookie named pf-accept-language. The cookie value must conform to guidelines defined under *IETF BCP 47* (tools.ietf.org/html/bcp47). For more information about the Java core method that PingFederate uses to parse the cookie, see *`Locale.forLanguageTag(String languageTag)`* (docs.oracle.com/se/7/docs/api/java/util/Locale.html#forLanguageTag(java.lang.String)).

Note that this locale-override behavior is the default implementation of the Java interface `LocaleOverrideService` defined in the PingFederate SDK (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 via the `LanguagePackMessages` class available in the PingFederate SDK. An instance of this class is passed into every template file and available for use there. For information, refer to 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
  - JMX
  - Connection Management
  - SSO Directory Service
- Passwords used by instances of the Simple Username Password Credential Validator
- Passwords used for encrypting certificates exported with their private keys
- Pass phrases used by IdP Discovery
- Passwords used by administrative console credentials when native authentication is used

Note: Passwords external to PingFederate—passwords used by instances of the Data Stores, for example—are not subject to this password policy.

Steps

1. Edit the `password-rules.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
2. Save the changes.

3. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node. No changes or restart of PingFederate is required on the engine nodes.

Managing cipher suites

About this task

The SSL/TLS server-client handshake involves negotiating cipher suites to be used for encryption and decryption on each side of a secured transaction. Cipher suites are stored 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`

These cipher-suite configuration files are located in the `<pf_install>/server/default/data/config-store` directory. Weaker cipher suites are commented out in these files. Retain this cipher-suite configuration to ensure the most secure transactions.

Important: Due to the import restrictions of some countries, Oracle Server JRE (Java SE Runtime Environment) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see the Java 8 release notes from Oracle (www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html).

For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide (docs.oracle.com/en/java/javase/11/migrate/).

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 may enable the same selection mechanism as well.

Steps

- To enable, disable, or re-order cipher suites, follow these steps.
  a. Edit the applicable cipher-suite configuration file.
  b. Save your changes.
  c. Restart PingFederate.

     For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on the **System#** **Cluster Management** screen.

     Important: For each engine node, restart PingFederate to load the changes made in the cipher-suite configuration file after the configuration is replicated.
- To enable cipher-suite selection based on listing order after an upgrade, follow these steps.
  a. 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>
```

b. Save this file as `cipher-suite-settings.xml` in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
c. Restart PingFederate.

   For a clustered PingFederate environment, perform these steps on the console node, and then click **Replicate Configuration** on the **System# Cluster Management** screen.

> **Important:** For each engine node, restart PingFederate to load the changes made in the `cipher-suite-settings.xml` file after the configuration is replicated.

Managing externally stored authentication sessions

Authentication sessions control when previously authenticated users are redirected back to the authentication sources on subsequent requests for browser-based SSO and PingFederate user-facing applications.

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 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 PingDirectory server, configure a cleanup plug-in in PingDirectory to suit the needs of your organization.

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. We recommend making changes gradually to observe the impact, if any.

> **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.
   a. Edit the `timer-intervals.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
   b. Update the StoredSessionCleanerInterval value (in milliseconds).

      The default value is `86400000`, which is 24 hours.
   c. Save your change.

2. Optional: Configure other cleanup options.

   a. Edit the `org.sourceid.saml20.service.session.data.impl.SessionStorageManagerJdbcImpl.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   Refer to 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`.

   b. Save your change.

3. If you have made any changes, restart PingFederate.

   In a clustered PingFederate environment, no changes or restart of PingFederate is required on any of the engine nodes.

Managing authentication sessions stored in PingDirectory

When storing persistent authentication sessions on a PingDirectory™ server, you must also configure a cleanup plug-in 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.

   a. Edit the `<pf_install>/pingfederate/server/default/data/config-store/timer-intervals.xml` file.
   b. Update the StoredSessionCleanerInterval value to `0`.
   c. Save your changes.
   d. Restart PingFederate.
2. Sign on to the PingDirectory administrative console.
3. Go to the **Configuration**# **Plug-in Root** screen.
4. Click **New Plugin** and then select **Purge Expired Data Plugin**.
5. Configure a new instance of the **Purge Expired Data Plugin**.

   Refer to the following table for information about each required field.

Field	Description
Name	The name of this plug-in instance.
Enabled	The status of this plug-in instance. Select the check box to enable this plug-in instance. Clear the check box to disable this plug-in instance. This check box is not selected by default.

Field	Description
Datetime Attribute	The attribute of which its value determines whether an authentication session has expired in the context of this plug-in 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 plug-in instance should only remove persistent authentication sessions that have expired. This plug-in 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 plug-in 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. The field has no default value.
Purge Behavior	The method how this plug-in instance remove expired data. Select **subtree-delete-entries** from the list. This field has no default selection.
Polling Interval	The frequency of which this plug-in instance should be run. Enter an integer to indicate the time value, followed by its unit of measurement. The field has no default value.

Field	Description
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 **subtree-delete-entries** is the selected from the **Purge Behavior** list, deletion of the entire subtree is considered a single update for the purposes of throttling. The field has no default value.

6. Click **Save**.

OAuth persistent grants cleanup

Authorization grants obtained by OAuth clients in the following manners are considered persistent.

- Grants obtained or updated by using the **Authorization Code**, **Resource Owner Credentials**, or **Device Authorization** grant type, in conjunction with the **Refresh Token** grant type.

  If the use cases involve mapping attributes from authentication sources (IdP adapter instances or IdP connections) or Password Credential Validator (PCV) instances to the access tokens (directly or through persistent grant extended attributes), such attributes and their values are stored along with the persistent grants so that they can be reused 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.

  If the use cases involve mapping attributes from authentication sources or PCV instances to the access tokens (directly or through persistent grant extended attributes), attribute values are obtained at runtime for each token request. No attributes or their values are stored with the persistent grants.

Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked or cleaned up. 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.

Manage expired persistent grants

About this task

PingFederate removes expired persistent grants once a day. The cleanup task removes 500 expired grants at a time until all expired grants are removed. 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 (or both) adds more workload to your storage server. We recommend making changes gradually to observe the impact, if any.

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 (version 7.0 or a more recent version), you have the option to use the PingFederate cleanup task or configure a cleanup plug-in in PingDirectory instead. The plug-in allows fine-grained control over various aspects of the cleanup task, which could smooth out the performance impact. For more information and configuration steps, see *Manage expired persistent grants in PingDirectory* on page 283.

Steps

1. Optional: Adjust the frequency of the cleanup task.
   a. Edit the `timer-intervals.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
   b. Update the AccessGrantCleanerInterval value (in milliseconds).

      The default value is `86400000`, which is 24 hours.
   c. Save your change.
2. Optional: Adjust the number of expired grants to be removed per batch.
   a. 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.AccessGrantManagerJ`
PingDirectory	`org.sourceid.oauth20.token.AccessGrantManagerL`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerL`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerL`

   b. Update the ExpiredGrantBatchSize value.

      The following example shows an updated value of 400.

```
<?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`.
   c. Save your change.
3. If you have made any changes, restart PingFederate.

   In a clustered PingFederate environment, no changes or restart of PingFederate is required on any of the engine nodes.

Manage expired persistent grants in PingDirectory

About this task

When storing OAuth persistent grants on a PingDirectory™ server (version 7.0 or a more recent version), you can configure a cleanup plug-in in PingDirectory to remove expired data from your directory server. This PingDirectory plug-in allows fine-grained control over various aspects of the cleanup task. For example, you can configure the maximum number of updates per seconds to smooth out 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.

   a. Edit the `timer-intervals.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
   b. Update the AccessGrantCleanerInterval value to `0` (zero).
   c. Save your change.
   d. Restart PingFederate.
2. Configure an instance of the PingDirectory plug-in to clean up expired data.
   a. Sign on to the PingDirectory administrative console.
   b. Go to the **Configuration#** **Plug-in Root** screen.
   c. Click **New Plugin** and then select **Clean up Expired PingFederate Persistent Access Grants Plugin**.
   d. Configure a new instance of the **Clean up Expired PingFederate Persistent Access Grants Plugin**.

      Refer to the following table for information about each required field.

Field	Description
Name	The name of this plug-in instance.
Enabled	The status of this plug-in instance. Select the check box to enable this plug-in instance. Clear the check box to disable this plug-in 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 plug-in 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 comfortably above the average rate that expired data is generated. The default value is `100`.

   e. Click **Save**.

Manage cleanup of persistent grants

About this task

Starting with version 9.2, PingFederate is capable of capping the number of persistent grants based on a combination of user, client, grant type, and authentication context. It 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 works 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.

It is worth mentioning that this cleanup runs on every engine node in a clustered PingFederate environment. Additionally, it does not replace the cleanup task or the PingDirectory™ plug-in engineered to manage expired grants. Working in tandem, they keep the size of the grant data store 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, this cleanup task is disabled. You may 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.AccessGrantManagerJd`
PingDirectory™	`org.sourceid.oauth20.token.AccessGrantManagerLD`
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLD`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerLD`

2. Look 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 may use the default values based on the inline comment. You may 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 the **System**# **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Specify the domain of the PF cookie

About this task

PingFederate identifies sessions by their respective PF cookie. By default, the PF cookie is set without domain information in the HTTP header; for example:

```
Set-Cookie: PF=zOv4xxmzDI2rx1TFBFy78X;Path=/;Secure;HttpOnly
```

As needed, you may configure PingFederate to return the Set-Cookie HTTP header with domain information.

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-domain element; for 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, and then click **Replicate Configuration** on the **System**# **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Results

Once this change is activated, PingFederate includes domain information in its Set-Cookie HTTP header; for example:

```
Set-Cookie:
PF=aDfPx6uwbbWGFhwE6zEhEG;Path=/;Domain=.example.com;Secure;HttpOnly
```

Specify the domain of the PF.PERSISTENT cookie

About this task

PingFederate identifies persistent authentication sessions by their respective PF.PERSISTENT cookie. By default, the PF.PERSISTENT cookie is set without domain information in the HTTP header; for example:

```
Set-Cookie: PF.PERSISTENT=UoBlPlf16V2oYAEPot2DnpUOXxitK7au;Path=/;Expires=Sat,
06-Nov-2021 00:48:08 GMT;Max-Age=94608000;Secure;HttpOnly
```

As needed, you may configure PingFederate to return the Set-Cookie HTTP header with domain information.

Steps

1. Edit the `persistent-session-cookie-config.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
2. Modify the cookie-domain element; for 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, and then click **Replicate Configuration** on the **System**# **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Results

Once this change is activated, PingFederate includes domain information in its Set-Cookie HTTP header; for example:

```
Set-Cookie:
PF.PERSISTENT=tOYwPM7VFMeluUyeu0EKQLL0DCJyVOqG;Path=/;Domain=.example.com;Expires=Sat,
06-Nov-2021 01:00:34 GMT;Max-Age=94608000;Secure;HttpOnly
```

Extend the lifetime of the PF cookie

About this task

PingFederate identifies sessions by their respective PF cookies. Some adapters, such as the HTML Form Adapter, also utilize the PF cookie to manage their adapter-sessions.

By default, the PF cookie is a session cookie. You may extend the lifetime of the PF 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* on page 372.

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 PF 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 the **System**# **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Configure forward proxy server settings

About this task

You can configure PingFederate to send web traffic (HTTP, HTTPS, or both) that it initiates through a forward proxy server.

Steps

1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
2. Look for the following properties:

```
#http.proxyHost=<HTTP_PROXY_HOST>
#http.proxyPort=<HTTP_PROXY_PORT>
#https.proxyHost=<HTTPS_PROXY_HOST>
#https.proxyPort=<HTTPS_PROXY_PORT>
#http.nonProxyHosts=*.internal.com|localhost
```

3. Optional: Configure forward proxy server settings for HTTP traffic.
   a. Remove the number sign (`#`) in front of `http.proxyHost` and `http.proxyPort`.
   b. Enter the hostname or the IP address of the forward proxy server.
4. Optional: Configure forward proxy server settings for HTTPS traffic.
   a. Remove the number sign (`#`) in front of `https.proxyHost` and `https.proxyPort`.
   b. Enter the hostname or the IP address of the forward proxy server.
5. Optional: Configure an exclusion list.
   a. Remove the number sign (`#`) in front of `http.nonProxyHosts`.
   b. 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; for example:

      ```
      *.example.com|localhost
      ```
6. Save your changes.
7. Restart PingFederate.

   For a clustered PingFederate environment, repeat these steps on each node.

Add custom HTTP response headers

About this task

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 for Microsoft Internet Explorer interoperability.

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 the **System# Cluster Management** screen.

   Important: For each engine node, restart PingFederate to load the changes made in the `response-header-admin-config.xml` or `response-header-runtime-config.xml` file (or both) after the configuration is replicated.

Configure validation for the AudienceRestriction element

About this task

For any IdP connections 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.

As needed, you may 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. For each IdP connection that you want to disregard the validation condition, add its **Partner's Entity ID** value as an entry inside the c:map element; for 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>
```

   In this example, the first entry adds the IdP connection with a **Partner's Entity ID** of `idp.example.com` to the list 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 the **System**# **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Customize the OpenID Provider configuration endpoint response

About this task

The OpenID Provider (OP) configuration endpoint at `/.well-known/openid-configuration` provides configuration information for the OAuth clients to interface with PingFederate using the OpenID Connect protocol.

As needed, you can customize the amount of configuration information by modifying a template file. Further, you may add conditional statements to return different responses based on information from the requests to suit multiple use cases simultaneously.

Steps

1. Edit the `openid-configuration.template.json` file, located in the `<pf_install>/pingfederate/server/default/conf/template` directory, to specify the desired information to be returned by the OpenID Provider 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 does not require a restart of PingFederate. For a clustered PingFederate environment, repeat these steps on each node.

Customize the heartbeat message

About this task

The heartbeat endpoint (`/pf/heartbeat.ping`) returns an "OK" browser message and an HTTP 200 status indication if the PingFederate server is running. You can customize the message by modifying a PingFederate property and a Velocity template file.

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 to `true` in the `<pf_install>/pingfederate/bin/run.properties` file.
2. Restart PingFederate.
3. Edit the `heartbeat.page.template` template file, located in the `<pf_install>/pingfederate/server/default/conf/template` directory, to specify the desired information to be returned by the heartbeat endpoint.

   An inline sample is provided.
4. Save your changes.

   Template customization does not require a restart of PingFederate. For a clustered PingFederate environment, repeat these steps on each node.

Customize the favicon for application and protocol endpoints

About this task

PingFederate provides a favorite icon (favicon) for its application (such as `/idp/startSSO.ping`) and protocol endpoints (for example, `/idp/SSO.saml2`).

Steps

1. Replace the `favicon.ico` file in the `<pf_install>/pingfederate/server/default/conf/template/assets/images` directory.
2. Restart PingFederate.

   For a clustered PingFederate environment, repeat these steps on each node.

Configure the behavior of searching multiple data stores with one mapping

About this task

Starting with PingFederate 8.1, if a data store 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 data store in the attribute source setup. This default behavior applies to all lookup configurations using multiple data stores in one mapping (see *Attribute mapping with multiple data sources* on page 870).

If you prefer PingFederate to abort the request immediately, which is the default behavior of PingFederate 8.0 and earlier versions, you can override the behavior by modifying a configuration file. Like the default behavior, this override also applies to all lookup configurations using multiple data stores 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.

   Create this file if it does not exist.
2. To override the default behavior, change the value of the AbortOnAttrLookupFailure element from `false` (the default value) to `true`.

   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 `<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 the **System# Cluster Management** screen.

Expected result when this override is set

If a data store uses results from previous queries as input, and if the previous queries return no result, PingFederate records an error message in the server log, abort the request immediately, and returns an error message to the user, the application, or the partner.

Security management

The **Security** menu provides access to security and infrastructure-related settings. Depending on the setup of PingFederate, menu items vary. Menu items are organized into two groups: **Certificate & Key Management** and **System Integration**.

Certificate and key management

About this task

The PingFederate administrative console provides a suite of configuration wizards for administrators to manage keys and certificates for various purposes. Tasks includes:

- Manage trusted certificate authorities (CAs)
- Manage server certificates for the administrative port and runtime ports
- Manage client certificate for mutual TLS authentication
- Manage signing and decryption keys and certificates
- Manage OAuth and OpenID Connect keys
- Manage certificates from partners
- Configure certificate revocation settings
- Manage partner metadata URLs
- Rotate system keys

For optimal security, PingFederate can be configured to use a hardware security module (HSM) for cryptographic material storage and operations. Standards such as the Federal Information Processing Standard (FIPS) 140-2 require the storage and processing of all keys and certificates on a certified cryptographic module.

Note: Management of keys and certificates is restricted to administrative users with the **Crypto Admin** administrative role (see *Administrative accounts* on page 141).

Refer to subsequent topics for configuration steps.

Manage trusted certificate authorities

You can import your federation partner's CA certificate or self-signed certificates into PingFederate's global trust list on the **Security# Trusted CAs** screen. If the certificate authority (CA) is not one of the major authorities, you may 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, it is not necessary to import the same certificate into the PingFederate store.

Importing a certificate

1. On the **Trusted CAs** screen, click **Import**.
2. On the **Import Certificate** screen, choose the applicable certificate file.

   If PingFederate is integrated with an 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** screen, review your configuration, amend as needed, and click **Save**.

Exporting a certificate

1. On the **Trusted CAs** screen, select **Export** under **Action** for the certificate.
2. On the **Export Certificate** screen, click **Next**.
3. On the **Export & Summary** screen, click **Export** to save the certificate file and then click **Done**.

Reviewing a certificate

1. On the **Trusted CAs** screen, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.

   When finished, close the pop-up window.

Removing a certificate

1. On the **Trusted CAs** screen, select **Delete** under **Action** for the certificate.

   To cancel the removal request, select **Undelete** under **Action** for the certificate.
2. Click **Save** to confirm your action.

Manage SSL server certificates

Use the **Security# SSL Server Certificates** screen to 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, they can be activated (or deactivated) for the administrative console, the runtime server, or both. Additionally, any of them may be selected as the new default certificate for the administrative console, the runtime server, or both at a latter time.

When creating a certificate, additional domain names may be added 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 SNI (Server Name Indication) 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. Consider the following sample configuration and inbound requests.

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; therefore, 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.
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.

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.

Creating a new certificate

1. On the **SSL Server Certificates** screen, click **Create new**.

2. On the Create Certificate screen, 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 (C) where the company is based.
Validity (days)	The time during which the certificate is valid.
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** screen, review your configuration, amend as needed, and click **Save**.

Importing a certificate and its private key

1. On the **SSL Server Certificates** screen, click **Import**.
2. On the **Import Certificate** screen, choose the applicable certificate file and enter its password.

   Note:

   If PingFederate is integrated with an HSM from Thales, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate. You must select a certificate that uses the RSA key algorithm.

   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** screen, review your configuration, amend as needed, and click **Save**.

Creating a certificate-authority signing request (CSR)

1. On the **SSL Server Certificates** screen, select **Certificate Signing** under **Action** 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 screen 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** screen, select the **Generate CSR** option.
3. On the **Generate CSR** screen, click **Export** to save the CSR file, and then click **Done**.

   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)

1. On the **SSL Server Certificates** screen, select **Certificate Signing** under **Action** for the certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option.
3. On the **Import CSR Response** screen, choose the applicable CSR response file.
4. On the **Summary** screen, review your configuration, and click **Save**.

Exporting a certificate

1. On the **SSL Server Certificates** screen, select **Export** under **Action** for the certificate.
2. On the **Export Certificate** screen, 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.

     CAUTION:

     This export contains the private key of the certificate. You must also enter an encryption password.

     If the selected certificate is stored in an HSM, the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file, and then click **Done**.

Reviewing a certificate

1. On the **SSL Server Certificates** screen, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.

   When finished, close the pop-up window.

Activating or deactivating a certificate

1. On the **SSL Server Certificates** screen, 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

1. On the **SSL Server Certificates** screen, select **Delete** under **Action** for the certificate.

   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 **Undelete** under **Action** for the certificate.
2. Click **Save** to confirm your action.

Manage SSL client keys and certificates

On the **Security# SSL Client Keys & Certificates** screen, you create and manage your authentication private keys and the certificates your server presents as a client in an outbound SSL/TLS transaction.

Creating a new certificate

1. On the **SSL Server Certificates** screen, click **Create new**.
2. On the **Create Certificate** screen, 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 (C) where the company is based.
Validity (days)	The time during which the certificate is valid.
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** screen, review your configuration, amend as needed, and click **Save**.

Importing a certificate and its private key

1. On the **SSL Client Keys & Certificates** screen, click **Import**.

2. On the **Import Certificate** screen, 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** screen, review your configuration, amend as needed, and click **Save**.

Creating a certificate-authority signing request (CSR)

1. On the **SSL Client Keys & Certificates** screen, select **Certificate Signing** under **Action** 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 screen 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** screen, select the **Generate CSR** option.
3. On the **Generate CSR** screen, click **Export** to save the CSR file, and then click **Done**.

   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)

1. On the **SSL Client Keys & Certificates** screen, select **Certificate Signing** under **Action** for the certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option.
3. On the **Import CSR Response** screen, choose the applicable CSR response file.
4. On the **Summary** screen, review your configuration, and click **Save**.

Exporting a certificate

1. On the **SSL Client Keys & Certificates** screen, select **Export** under **Action** for the certificate.
2. On the **Export Certificate** screen, 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.

     > CAUTION**:**
     >
     > This export contains the private key of the certificate. You must also enter an encryption password.

     If the selected certificate is stored in an HSM, the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file, and then click **Done**.

Reviewing a certificate

1. On the **SSL Client Keys & Certificates** screen, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.

   When finished, close the pop-up window.

Removing a certificate

1. On the **SSL Client Keys & Certificates** screen, select **Delete** under **Action** for the certificate.

   To cancel the removal request, select **Undelete** under **Action** for the certificate.
2. Click **Save** to confirm your action.

Manage digital signing certificates and decryption keys

On the **Security# Signing & Decryption Keys & Certificates** screen, you can create and maintain certificates (and thus their respective key pairs) for the purpose of signing outgoing requests, responses, assertions, and access tokens, and for the purpose of decryption.

CAUTION: For best security, we recommend using separate certificates for signing and decryption.

After creating your certificates, if they remain as self-signed certificates, you can optionally enable automatic certificate rotation.

Creating a new certificate

1. On the **Signing & Decryption Keys & Certificates** screen, click **Create new**.
2. On the **Create Certificate** screen, 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 (C) where the company is based.
Validity (days)	The time during which the certificate is valid.
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.

Field	Description
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** screen, review your configuration, amend as needed, and click **Save**.

Importing a certificate and its private key

1. On the **Signing & Decryption Keys & Certificates** screen, click **Import**.
2. On the **Import Certificate** screen, 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** screen, review your configuration, amend as needed, and click **Save**.

Creating a certificate-authority signing request (CSR)

1. On the **Signing & Decryption Keys & Certificates** screen, select **Certificate Signing** under **Action** 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 screen 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** screen, select the **Generate CSR** option.
3. On the **Generate CSR** screen, click **Export** to save the CSR file, and then click **Done**.

   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)

1. On the **Signing & Decryption Keys & Certificates** screen, select **Certificate Signing** under **Action** for the certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option.
3. On the **Import CSR Response** screen, choose the applicable CSR response file.
4. On the **Summary** screen, review your configuration, and click **Save**.

Exporting a certificate

1. On the **Signing & Decryption Keys & Certificates** screen, select **Export** under **Action** for the certificate.

2. On the **Export Certificate** screen, 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.

     ⓘ **CAUTION:**

     This export contains the private key of the certificate. You must also enter an encryption password.

     If the selected certificate is stored in an HSM, the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file, and then click **Done**.

Reviewing a certificate

1. On the **Signing & Decryption Keys & Certificates** screen, select the certificate by its serial number.
2. Review the selected certificate in the pop-up window.

   When finished, close the pop-up window.

Reviewing a certificate's usage

1. On the **Signing & Decryption Keys & Certificates** screen, select **Check Usage** under **Action** for the certificate.

   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.

   When finished, close the pop-up window.

Removing a certificate

1. On the **Signing & Decryption Keys & Certificates** screen, select **Delete** under **Action** for the certificate.

   To cancel the removal request, select **Undelete** under **Action** for the certificate.
2. Click **Save** to confirm your action.

Certificate rotation

PingFederate supports automatic certificate rotation for self-signed certificates created for the purpose of signing SAML requests, responses, and assertions, or XML decryption for Browser SSO and WS-Trust STS transactions on a per-certificate basis. This optional feature greatly reduces the cost of managing self-signed certificates.

ⓘ **Note:** Certificate rotation is only available to self-signed certificates.

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 twenty-five and ten percent of the original lifetime of the current certificate. 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 the Creation Buffer field	The default value for the Activation Buffer field	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 shutdown when the **Creation Buffer** threshold is reached for a given certificate, a new key pair and a new certificate is 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 the **System**# **Cluster Management** screen.

Although optional, it is recommended that you turn on notifications for certificate events in the **System**# **Runtime Notifications** screen. When configured, PingFederate notifies the configured recipient when a new certificate becomes available and when it is activated. Depending on the role of the certificate, you can update your partner accordingly.

Connection and federation metadata

Certification rotation is a per-certificate configuration. When certificate rotation is enabled for a certificate and a new certificate using a new key pairs becomes available, PingFederate deploys the new certificate to all enabled connections using that certificate. The actions taken by PingFederate vary depending on the role of the certificate.

Notifications

Although optional, it is recommended that you turn on notifications for certificate events in the **System**# **Runtime Notifications** screen. When configured, PingFederate notifies the configured recipient when a new certificate becomes 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 Browser 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 becomes available, PingFederate performs the following tasks for all SAML 2.0 connections using the same decryption key:

- Push the current decryption key from primary to secondary.
- Place the new certificate as the primary decryption key.
- Update the decryption key with the new certificate in the metadata.

- Start 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 supports the RSA key algorithm only. When **EC** (elliptic curve) is selected as the **Key Algorithm** value on the **Certificate Rotation** screen, 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 for you before the **Activation Buffer** threshold is reached.

Federation metadata for Browser SSO connections

PingFederate updates the metadata for the applicable Browser SSO connections as soon as a new certificate becomes available.

To ensure that your partners are aware of the new certificate, you can provide the partners their respective federation metadata URL or metadata export.

Metadata by URL

PingFederate runtime engine provides an endpoint (`/pf/federation_metadata.ping`) to return metadata for Browser SSO connections. An SP or an 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 that 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 the **System**# **Cluster Management** screen when the new certificate becomes available, such that the federation metadata for these connections are 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 **Manage All** connections management screen or the **System**# **Metadata Export** wizard.

> ⓘ **Note:** PingFederate does not deploy new certificates or update metadata for inactive connections.

WS-Trust STS connections

For connections with only the WS-Trust 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 Browser SSO profile. Your partner can reuse the certificate from the metadata (by URL or manual export) and apply it to its STS configuration.

Manage certificate rotation settings

About this task

You manage certificate rotation settings for self-signed certificates in the **Security**# **Signing & Decryption Keys & Certificates** screen.

Steps

1. On the **Certificate Management** screen, select **Certificate Rotation** under **Action** for the applicable certificate.

   > ⓘ **Note:** Certificate rotation is only available to self-signed certificates.

2. On the **Enable Certificate Rotation** screen, select the check box to turn on certificate rotation for the selected certificate.

   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** screen, 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 supports the RSA key algorithm only. When **EC** (elliptic curve) is selected as the **Key Algorithm** value on the **Certificate Rotation** screen, PingFederate does not update the SAML 2.0 connections and their metadata.

Field	Description
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-SHA256, SHA384 and SHA512; and ECDSA-SHA256, SHA384 and SHA512.) The default value matches that of the current certificate.

4. On the **Certificate Rotation Summary** screen, review the rotation settings. Adjust as needed or click **Save** to turn on automatic certificate rotation for this certificate.

Managed SP connection to PingOne for Enterprise and signing certificate

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 as part of the initial setup or the **System**# **Connect to PingOne for Enterprise** configuration wizard in PingFederate 8.0 (or a more recent release).

The certificate rotation settings are as follow:

Field	Values
Creation Buffer (days)	90
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.

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* on page 163).

Manage keys for OAuth and OpenID Connect

About this task

On the **Security**# **OAuth & OpenID Connect Keys** screen, specify whether PingFederate should use static or dynamically rotating keys for OAuth and OpenID Connect. These keys are used to in the following manner:

PingFederate role	Key usages
OpenID Provider (OP)	Sign ID tokens for RPs
Relying Party (RP)	Sign JWTs for authentication, sign OpenID Connect request objects, decrypt ID tokens, or a any combination of them.

Configure static signing keys

About this task

You can specify whether PingFederate should use static or dynamically rotating keys to sign ID tokens, JWTs for client authentication, and OpenID Connect request objects.

Steps

1. Go to the **Security# OAuth & OpenID Connect Keys** screen.
2. Select the **Enable Static Keys** check box to use static keys for OAuth and OpenID Connect.

   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.

   Once selected, the administrative console displays the following fields under **Signing 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	Required	Optional	Optional

3. Follow these steps to complete the configuration under **Signing Keys**.
   a. For the RSA key type, select an active signing key and optionally a previous signing key.

      If the desired signing key is not found, click **Manage Certificates** to create it.

      There is no default selection.

      The active signing key and the previous signing key (if configured) are published at the PingFederate JSON Web Key (JWK) Set endpoint `/pf/JWKS`.
   b. For each applicable EC (elliptic curve) key type, select an active signing key and optionally a previous signing key.

      If the desired signing key is not found, click **Manage Certificates** to create it. Alternatively, complete the configuration, create the desired signing later keys later, and then update the configuration afterward.

      There is no default selection.

      The active signing key and the previous signing key (if configured) are published at the PingFederate JWKS endpoint `/pf/JWKS`.
   c. Optional: For any key type that 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**.

Results

ⓘ **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 not returned by the PingFederate JWKS endpoint `/pf/JWKS`. Signing algorithms associated with EC key types that have not been configured with an active static signing key are hidden.

For existing clients and 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 involves that signing algorithm will fail. When you revisit the configuration, the administrative console displays an error message. Your options are described 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** screen and the default setting configured for all clients created via the Dynamic Client Registration protocol on the **Client Configuration Defaults** screen.

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. Note that runtime errors persist until the configuration issue is resolved.

These options are applicable to individual OpenID Connect IdP connections on the **OpenID Provider Info** screen.

Configure static decryption keys

About this task

You can specify whether PingFederate should use static or dynamically rotating keys to decrypt asymmetrically-encrypted ID tokens.

Steps

1. Go to the **Security# OAuth & OpenID Connect Keys** screen.
2. Select the **Enable Static Keys** check box to use static keys for OAuth and OpenID Connect.

   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.

   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

Key Type	**Active**	**Previous**	**Publish Certificate**
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 complete the configuration under **Decryption Keys**.

a. For each applicable key type, select an active decryption key and optionally a previous decryption key.

   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.

   The active decryption key is published at the PingFederate JSON Web Key (JWK) Set endpoint `/pf/JWKS`.

b. Optional: For any key type that 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: For each applicable decryption key, its associated chain of certificates is published as the x5c parameter value.

   The **Publish Certificate** check boxes are not selected by default.

Note: When static keys are enabled, you must also select an active *signing* key for the RSA key type.

4. Under **Signing Keys**, select an active key for the RSA key type.

If the desired key is not found, click **Manage Certificates** to create it.

There is no default selection.

The active signing key is published at the PingFederate JWKS endpoint `/pf/JWKS`.

5. Click **Save**.

Results

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 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":
 "hvHfiamhV4wGC9JHppJZjdKG5K3MvhWwo6PBsSQowGOTeILAbzO8Jfmp7nRxuujTE6k83RXNeWUvTwamGqShXv
vjoNZD8Cv0Y9C3R4Ckj6dBL70Osk_NfBR7MYmRA6dV0PJ5k4Lt_vQveXMkylD9XuLFP-
gqooMXkB6FCCLqZZAi0voi3WQ7ECzSta3ke9F5VFl7-4zVjRtJHjM9gGEhd5OkaZioqs9xBHeOrwhPbiPTsIA7ve
dpl_dKixFTdQYIBMmIWGUyuB43XYq106z9CWoOcw",
      "e": "AQAB"
    },
    ...
  ]
}
```

When static keys are used, the PingFederate JWKS endpoint `/pf/JWKS` returns only the configured active key (or 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":
 "AdvUA2YD2kn7COLkFIG2vL2k34CMv7VPxsvbgOJBL2exSziMGPw6YJp2eafuHlBom7bkjv3iFy5dTuGB7B28Zc
      "crv": "P-521"
      },
      ...
   ]
}
```

Manage certificates from partners

About this task

You receive certificates from partners for signature verification, encryption, and back-channel authentication. They are managed within connections.

Signature verification

You specify one or more certificates that PingFederate can use to validate the digital signatures found in inbound messages from your partners.

To manage such certificates for a given connection:

1. Select the connection to reach its **Activation & Summary** screen.
2. Select **Signature Verification Certificate**.
3. Click **Manage Certificates**.

   You can import, export, review, activate, deactivate, and remove certificates for signature verification on the **Certificate Management** screen.

Encryption

You specify a certificate that PingFederate uses to encrypt the outbound messages before delivering them to your partners.

To manage such certificates for a given connection:

1. Select the connection to reach its **Activation & Summary** screen.
2. Select **Select XML Encryption Certificate**.
3. Click **Manage Certificates**.

   You can import, export, review, activate, and remove certificates for encryption on the **Certificate Management** screen.

Back-channel authentication

You specify a certificate that Pingfederate uses to authenticate inbound (SOAP) messages from your partners by their client certificates.

To manage such certificates for a given connection:

1. Select the connection to reach its **Activation & Summary** screen.
2. Select **SSL Verification Certificate**.
3. Click **Manage Certificates**.

   You can import, export, review, activate, and remove certificates for back-channel authentication on the **Certificate Management** screen.

Note: Depending on the use cases, your connection to the partner may not require signature verification, encryption, inbound (SOAP) back-channel authentication by client certificate, or any such combinations. If so, the **Activation & Summary** screen does not display the related administrative screen.

Configure certificate revocation

About this task

By default at runtime, PingFederate attempts to retrieve a CRL to verify that a signing certificate has not been revoked, whenever a CRL distribution-point URL is included within the certificate. Optionally, on the **Security# Certificate Revocation Checking** screen, you can enable and configure OCSP checking as the preferred verification method, depending on your requirements. (For more information, see *Certificate validation* on page 116.)

OCSP can be used in place of CRL checking, or CRLs can be retained as a backup method (for failover).

Note: When OCSP is enabled, CRL checking is not done independently—only as a failover option for one or more OCSP failure conditions.

Steps

1. Optional: Configure OCSP.

For more information about each field, refer to the following table.

Field	Description
Enable OCSP	Turns on OCSP certificate-revocation checking. OCSP checking is not enabled by default.
Default OCSP Responder URL	The location of a URL to use for certificate-revocation checking, a backup used only if the OCSP Responder URL is not contained in the certificate.
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, as needed.
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,” below). 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 may 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	For the response to be considered 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)

Field	**Description**
Next Update Grace Period	If the response includes a `<nextUpdate>` timestamp indicating when updated certificate statuses will be 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	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)
Certificate is Unknown	The certificate does not fall under the purview of the CA associated with the OCSP Responder. The choices indicate whether an unknown certificate is to be considered valid or not, or whether to try CRL checking. The default selection is **Treat as Revoked**.
OCSP Responder is Unavailable	Indicates what action to take if the Responder cannot be reached. The choices indicate whether an unknown certificate is to be considered valid or not, 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 to be considered valid or not, or whether to try CRL checking. The default selection is **Treat as Revoked**.
Proxy Settings	If OCSP messaging 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 CRL checking, when CRL is enabled for failover.

2. Optional: Configure CRL checking.

For more information about each field, refer to 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 enabled 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.

Field/Selection	Description
Next Retry on Resolution Failure	Specifies the number of minutes the server waits before trying to retrieve a CRL when the previous attempt failed—applies only when the selection above (Treat Unretrievable CRLs as Revoked) is unchecked. The default value is `1440` (minutes, which is 24 hours)
Next Retry on Next Update Expiration	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 IdP adapter instance, reset the CRL cache. When it 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 (see *Manage trusted certificate authorities* on page 292). 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.

Transition to an HSM

About this task

Administrators may enable the HSM hybrid mode, which provides the choice to store each relevant key and certificate on a hardware security module (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 to mirror the setup.

When all relevant keys and certificates are stored on the HSM, administrators may 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* on page 90)

   ⓘ **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.

   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 may 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 that are stored on your HSM and the local trust store.

> **Important:** When making changes to keys and certificates, you may 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 the **Security#** **SSL Server Certificates** page.

   You may 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* on page 293.
3. Create new digital signing certificates and decryption keys on the **Security#** **Signing & Decryption Keys & Certificates** page and reconfigure connections or configuration items to use the new certificates and keys from your HSM.

   > **Tip:** You may use **Check Usage** to locate the applicable connections or configuration items.

   For configuration steps, see *Manage digital signing certificates and decryption keys* on page 299.
4. If your connections support outbound (SOAP) back-channel authentication by client certificates, create new SSL client certificates on the **Security#** **SSL Client Keys & Certificates** screen and reconfigure connections to use the new certificates from your HSM.

   > **Tip:** You may use **Check Usage** to locate the applicable connections or configuration items.

   For configuration steps, see *Manage SSL client keys and certificates* on page 297.
5. If you are transitioning to an nCipher HSM, export the trusted CA certificates from the local trust store and import them to your HSM on the **Security#** **Trusted CAs** screen and reconfigure configuration items to use the new certificates and keys from your HSM.

   > **Tip:** You may use **Check Usage** to locate the applicable configuration items.

   For configuration steps, see *Manage trusted certificate authorities* on page 292.
6. If you are transitioning to an nCipher 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 *Manage certificates from partners* on page 310.

Manage Partner metadata URLs

SAML metadata URL streamlines the processes of establishing and maintaining SAML connections. If your partner provides SAML metadata by URL, you may use the metadata URL for the following scenarios:

- Create a new SAML connection using the metadata URL and associate the metadata URL with the new connection.
- Enable or disable automatic update from the associated metadata URL.
- Add or update the metadata URL associated with an existing SAML connection.
- Update 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 subsequently for any of the aforementioned scenarios, PingFederate validates the digital signature using the previously stored certificate. If the signature is correct, the process carries on. If there is a digital signature error, PingFederate aborts the process and provides an error with a recommended course of action. As needed, the signature verification process can be bypassed.

Use the **Security# Partner Metadata URLs** screen to add, update, review, or remove SAML metadata URLs provided by your partners.

Adding a new metadata URL

1. On the **Partner Metadata URLs** screen, click **Add New URL**.
2. On the **URL** screen, define the metadata URL.
   a. Configure each field.

Field	Description
Name	A friendly 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 signature. Clear the check box to skip the signature verification process. This check box is selected by default.

   b. Click **Load Metadata**.
3. On the **Certificate Summary** screen, review the certificate information.

   Shown and applicable only when the **Validate Metadata Signature** check box on the **URL** screen 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** screen, review the configuration. Then, click **Done** and **Save**.

Updating an existing metadata URL

1. On the **Partner Metadata URLs** screen, select the applicable metadata by its name.
2. On the **URL** screen, update the name, URL, or digital signature verification option. Then, click **Next**.
3. On the **Certificate Summary** screen, 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.

   Shown and applicable only when the **Validate Metadata Signature** check box on the **URL** screen is selected.

   If the metadata is signed but the certificate is provided outside of the metadata, click **Import** to upload the verification certificate.
4. Click **Next**.

5. On the **Summary** screen, review the configuration, then click **Done** and **Save**.

Reviewing a metadata URL usage

1. On the **Partner Metadata URLs** screen, select **Check Usage** under **Action** for the applicable metadata.

   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.

   When finished, close the pop-up window.

Removing a metadata URL

1. On the **Partner Metadata URLs** screen, select **Delete** under **Action** for the applicable metadata.

   The **Delete** option is shown and applicable only when the metadata is not used by any connections.

   To cancel the removal request, select **Undelete** under **Action** for the certificate.
2. Click **Save** to confirm your action.

Rotate system keys

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 can ensure optimal security of your environment.

Steps

1. Go to the **Security**# **System Keys** screen.
2. To rotate the system keys, click **Rotate** and then **Save**.

System integration

If PingFederate plays the SP role, we recommend that you configure redirect validation rules to ensure valuable information, such as user attribute values, are 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. Finally, you can manage the availability and authentication requirements for the supporting services that PingFederate offers. These menu items are located under **Security**# **System Integration**.

Configure redirect validation

About this task

Several SP adapters can be configured 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 open the possibility that a third party (with specific knowledge of the IdP, the SP, or both; PingFederate endpoints; and PingFederate configuration) might be able to obtain and use valid security tokens to gain improper access to the target resource.

This potential security threat would involve 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 would be intended to intercept 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:

- The default target URL for any IdP connections (see *Configure default target URLs* on page 622)
- The default target URL for any adapter-to-adapter mappings (see *Configure a default target URL (optional)* on page 703)
- The SP default URL for successful SSO (see *Configure default URLs* on page 589)
- The IdP default URL for successful SLO (see *Configure a default URL and error message* on page 495)

They do *not* need to be entered into the list manually.

Furthermore, PingFederate is also capable of validating the error resource parameter. For more information about the InErrorResource parameter, see *IdP endpoints* on page 757, *SP endpoints* on page 761 and *System-services endpoints* on page 773.

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 were not selected in the previous PingFederate installation. Although optional, it is strongly recommended to enable validation for both target and error resources and to enter all expected resources (including the HTTPS option) to prevent unauthorized access.

Steps

1. On the **Security** menu, click **Redirect Validation** to open the **Local Redirect Validation** screen.
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.

   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.

   a. 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.

   b. Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

      Enter a value without the protocol; for example: `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.

   c. Optional: Enter the exact path of this resource under **Valid Path**.

      Starts 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 may allow specific query parameter (or 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.

   d. Optional: 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.

   e. 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.

   f. 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.

   g. Repeat these steps to define multiple expected resources.

      Note that 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**.

Manage partner redirect validation

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 via open redirections, PingFederate provides an option to validate wreply for 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, it is strongly recommended to enable wreply validation for SLO and to specify the allowed domains and paths for each WS-Federation connection to prevent unauthorized access.

Steps

1. On the **Security** menu, click **Redirect Validation** to open the **Partner Redirect Validation** screen.
2. Select the **Enable wreply Validation For SLO** check box to enable this feature.

   This check box is selected by default in new installations.

   Clear the check box to disable the feature.
3. Click **Save**.

Configure incoming proxy settings

When PingFederate is deployed behind a reverse proxy (or a similar network traffic management solution, such as a load-balancer), the options described in the following four sections enable PingFederate to use information in HTTP headers added by the reverse proxy to construct correct responses. These options, configurable on the **Security# Incoming Proxy Settings** screen, apply globally to all incoming requests.

Note:

These settings override Jetty proxy options.

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 reverse proxy or load balancer.

It is common for proxies to 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 into 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. It is common for proxies to 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 are using mutual client certificate authentication and would like 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** screen.

For example, suppose you have configured the Apache HTTP Server 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:

Since 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. In addition, the specified values must match the header names used in the Apache HTTP Server configuration (with the exception of 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 if HTTP is used between the reverse proxy and PingFederate.

Configure service authentication

About this task

If you are using the SAML 2.0 Attribute Query profile as an SP, then the requesting application(s) at your site must authenticate to the PingFederate server (see *Attribute Query and XASP* on page 45 and the */sp/startAttributeQuery.ping* on page 765 SP application endpoint).

In addition, authentication is required to access PingFederate runtime data via JMX (see *Runtime monitoring using JMX* on page 160) or to make SOAP calls to the Connection Management Service. Authentication is optional for the SSO Directory Service (see *Web service interfaces and APIs* on page 817).

Note: To help ensure network security, access to all of these services is deactivated when PingFederate is first installed.

On the **Security# Service Authentication** screen, administrators with the **Admin** administrative role can activate and configure authentication for Attribute Query, JMX, and SSO Directory.

To activate and configure authentication for the Connection Management Service, the administrators must be granted all three administrative roles: **Admin**, **Crypto**, and **User Admin**.

Steps

- Follow these steps to enable a service:
  a. Select **Activate** under **Action**.
  b. Enter (or modify) the service account name and define (or reset) the password.

     You and the application developer must agree to these values.

     Tip: Authentication is optional for the SSO Directory Service.
- To disable a service, select **Deactivate** under **Action**.

  Note: Although not accessible when deactivated, the Connection Management Service and the SSO Directory Service are still deployed by default as part of PingFederate. If your organization does not plan to use one or neither 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 prevents user accounts from becoming locked 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; for example, 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 have the options to 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.

Configure account lockout protection

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, refer to the inline comments and the following table.

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** on the **System# Cluster Management** screen.

Password spraying prevention

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 locked out for a time period. 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.

Configure password spraying prevention

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, refer to 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`.
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 the **System# Cluster Management** screen.

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 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.
- Refer to *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: `hivemodule.xml` and `com.pingidentity.crypto.jwk.MasterKeySet.xml`.

Steps

1. Stop PingFederate.
2. Open `<pf-install>/pingfederate/server/default/conf/META-INF/hivemodule.xml` in a text editor:

3. Scroll to the bottom of the file and locate the following lines:

```
<service-point id="MasterKeyEncryptor"
 interface="com.pingidentity.sdk.key.MasterKeyEncryptor">
    <create-instance
 class="com.pingidentity.crypto.jwk.NoOpMasterKeyEncryptor"/>
</service-point>
```

4. To enable master key encryption using AWS KMS, replace the lines shown in step 3 with the following lines:

```
<service-point id="MasterKeyEncryptor"
 interface="com.pingidentity.sdk.key.MasterKeyEncryptor">
    <create-instance
 class="com.pingidentity.pingcommons.aws.key.AwsKmsMasterKeyEncryptor"/>
</service-point>
```

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 this:

```
<?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.

Results
After configuring and starting PingFederate, the PingFederate master key file, `pf.jwk`, will be encrypted.

Authentication policies

Authentication policies, an optional configuration in PingFederate, help administrators implement complex authentication requirements. As needed, 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. Furthermore, administrators can reuse an authentication policy by ending it with an authentication policy contract or a local identity profile, and then apply the authentication policy contract in multiple use cases.

Selectors

Authentication selectors provide a plug-in capability for PingFederate to evaluate various conditions related to the requests. PingFederate comes bundled with a set of authentication selectors. for 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 IdP connections based on the PartnerIdpId parameter values provided in the 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 the `<pf_install>/pingfederate/sdk/doc` directory.

Managing authentication selector instances

About this task

You manage authentication selectors in the **Selectors#** **Manage Authentication Selector Instances** screen.

Steps

- 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**.

Note: You can only remove a selector instance if it is not deployed in any authentication policy.

Choosing a selector type

Steps

1. Enter a name and an ID for this authentication selector instance.
2. Select the desired type of authentication selector from the list.

Results

Configuring an authentication selector instance

About this task

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.
2. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Configuring the CIDR Authentication Selector

About this task

The CIDR Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the IP address of an incoming SSO request. 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 Adapter instances or between a Kerberos Adapter instance and an X.509 IdP Adapter instance. For example, use this selector in one or more authentication policies to route internal requests to a Kerberos Adapter instance.

Steps

1. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Add a new row to 'Networks'** and enter a network range. Then, click **Update**.

   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.

   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.

   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.
6. Optional: Enter a **Result Attribute Name** value.

   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 in order to use its value to fulfill an attribute contract or for issuance criteria.
7. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

policy paths: When you place this selector instance as a checkpoint in an authentication policy, it forms two **Yes** and **No**. If the IP address of an incoming 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

About this task

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 one or more authentication policies. 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. Click **Identity Provider**# **Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Next** to proceed to the next screen.
5. On the **Selector Result Values** screen, specify the relevant node index values.

   Each selector result value forms a policy path when you place this selector instance as a checkpoint in an authentication policy.

   a. Enter a node index value based on your cluster configuration under **Result Values** and click **Add**.

      This node index value should corresponds to a node index value of one of the engine nodes in the cluster.
   b. Optional: Add more values to differentiate criteria for authentication selection.

      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.
6. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Configuring the Connection Set Authentication Selector

About this task

The Connection Set Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the target SP connection used in an SSO request and SP connections configured within PingFederate. This selector allows you to override connection authentication selection on an individual connection basis in one or more authentication policies.

Steps

1. Click **Identity Provider**# **Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Add a new row to 'Connections'**, select an SP connection from the list, and click **Update**.

5. Optional: Repeat the previous step to add more connections.

   Display order does not matter.

   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.
6. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

policy paths: When you place this selector instance as a checkpoint in an authentication policy, it forms two **Yes** and **No**. 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

About this task

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. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, select a property from the **Extended Property** list.

   This 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 value or values defined in this selector instance.
5. On the **Selector Result Values** screen, specify one or more expected result values.
   a. Enter the exact (case-sensitive) value under **Result Values** and click **Add**.
   b. Optional: Add more values to differentiate criteria for authentication selection.

      Display order *could* 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; it exits and returns true as soon as it finds a match. 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.

As an example, suppose 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.

As an example, suppose the previous selector instance remains; the invoking connection is populated with an extended property values of `Alpha` and `Charlie`; and this selector instance remains as a checkpoint in an authentication policy as previously illustrated.

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`. Note that 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. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Example

1. On the **System# Extended Properties** screen, you defined a multivalued extended property; you named it configStatus.
2. You created an SP connection with the following characteristics:
   - On the **Extended Properties** screen, you added two values for the configStatus extended property: `DEV` and `TEST`.
   - On the **Attribute Source Mapping** screen, you mapped an authentication policy contract to the SP connection. The policy contract name is `APC`.

3. You created an instance of the Extended Property Authentication Selector with the following characteristics:
   - On the **Type** screen, you named the selector instance `ExProps`.
   - On the **Authentication Selector** screen, you selected configStatus from the list.
   - On the **Selector Result Values** screen, you enter `DEV` and `TEST`.
4. You created and activated the following IdP authentication policy:

```
ExtProps
+--DEV
|  OpenToken
|  +--Fail: Done
|  +--Success: APC
|
+--TEST
   HTML
   +--Fail: Done
   +--Success: APC
```

   You configured 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.

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: We do not recommend using this selector to determine whether, or not, an authentication source with a higher level of assurance should be bypassed because HTTP request headers could potentially be forged.

Steps

1. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Add a new row to 'Results'**, enter an expression for use when inspecting the HTTP header value of the target HTTP header under **Match Expression**, and click **Update**.

   Wildcard entries are allowed; for example, `*value*`.

5. Optional: Repeat the previous step to add more expressions.

   Display order does not matter.

   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.
6. Enter the type of HTTP header you want the selector to inspect in the **Header Name** field.

   This field is *not* case-sensitive.
7. Optional: Clear the **Case-Sensitive Matching** check box to disable case-sensitive matching between the HTTP header values from the requests and the **Match Expression** values specified on this screen.

   The **Case-Sensitive Matching** check box is selected by default.
8. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

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*`
Internet Explorer	`*MSIE*` ⓘ **Tip:** For more information, see *User-agent string changes* from Microsoft (msdn.microsoft.com/library/hh869301.aspx).
Safari	`*Safari*`

2. Enter `User-Agent` in the **Header Name** field.

Configuring the HTTP Request Parameter Authentication Selector

About this task

The HTTP Request Parameter Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on query parameter values. 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 SSO request.

ⓘ **Important:** We do not recommend using this selector to determine whether, or not, an authentication source with a higher level of assurance should be bypassed because query parameters could potentially be forged.

Steps

1. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, configure the applicable selector instance settings.
   a. 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 **Authentication Policies# Tracked HTTP Parameters** screen. For more information, see *Defining authentication policies* on page 342.

   b. Optional: Clear the **Case-Sensitive Matching** check box 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** screen.

      The **Case-Sensitive Matching** check box is selected by default.

   c. Optional: Enable policy paths to handle additional scenarios.

      For more information, refer to 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** screen, enter a request parameter value under **Result value** and click **Add**.

   Wildcard entries are allowed; for example, `*value*`.

   > ⓘ **Important:** A more specific match is considered a better match and an exact match is considered the best match.

6. Optional: Repeat the previous step to add more request parameter values.

   Display order does not matter.

   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. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Example

Suppose you enter three selector result values (`Central`, `Easter`, and `Southern`) on the **Selector Result Values** screen, as illustrated in the following screen capture.

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

About this task

The 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.

> ⓘ **Note:** The OAuth Client Set Authentication Selector is only applicable to OAuth clients using the authorization code or implicit flow.

Steps

1. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Add a new row to 'Clients'**, select an OAuth client from the list, and click **Update**.

5. Optional: Repeat the previous step to add more clients.

   Display order does not matter.

   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.
6. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

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

About this task

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).

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 rather than username and password.

Note: Configure one or more scopes in **OAuth Server**# **Authorization Server Settings** screen if you have not already done so.

Steps

1. Click **Identity Provider**# **Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, select the required scopes, scope groups, or both.

   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 SSO requests do not match this authentication selector's criteria and result in a returned result value of `No`. Therefore, if you are using this selector and selectors specific to SAML connections, place this selector first in the mapping list so that it takes precedence for OAuth without disrupting selector logic on SAML connections.
5. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

When you place 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 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**.

Configuring the Requested AuthN Context Authentication Selector

About this task

The Requested AuthN Context Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the authentication context (or contexts) requested by an SP for Browser SSO requests or an RP for OAuth with OpenID Connect use cases in one or more authentication policies.

For Browser SSO, this authentication selector works in conjunction with SP connections via SAML 2.0 only, using the SP-initiated SSO profile; other Browser SSO protocols do not support authentication context. For OAuth, clients supporting the OpenID Connect protocol must include the optional acr_values parameter in their authorization requests to indicate their preferred authentication context (or contexts).

Steps

1. Click **Identity Provider**# **Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, configure the applicable selector instance settings.
   a. 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** screen.

      When selected (the default), the check box on this screen provides a means of either:

      - Adding the value of the authentication context determined by the selector into the SAML assertion.
      - When applicable, replacing any value returned from the associated adapter instance with the selector-result value.

   b. 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. On the **Selector Result Values** screen, specify the authentication contexts to be used as the criteria.
   a. Enter the exact (case-sensitive) parameter value under **Result Values** and click **Add**.

      The value may include URIs defined in *Authentication Context for the OASIS Security Assertion Markup Language (SAML) 2.0* (docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf) or any other value agreed upon with the partner.
   b. 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 4b*).

      Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.
6. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Configuring the Session Authentication Selector

About this task

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. Click **Identity Provider# Selectors** to open the **Manage Authentication Selector Instances** screen.
2. On the **Manage Authentication Selector Instances** screen, click **Create New Instance** to start the **Create Authentication Selector Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authentication selector instance.
4. On the **Authentication Selector** screen, click **Add a new row to 'Authentication Sources'**, 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: Authentication sessions must be enabled for the selected authentication source (or globally for all authentication sources) on the **Sessions** screen. Click **Manage Sessions** to review and configure authentication sessions.
5. Optional: Repeat the previous step to add more authentication sources.

   Display order *could* 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.

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.

6. 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.
7. To complete the configuration:
   a. Click **Done** on the **Summary** screen.
   b. Click **Save** on the **Manage Authentication Selector Instances** screen.

Results

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.

When this selector instance (named Intranet sessions) is placed in a policy, four policy paths are formed.

Configuring a sample use case

About this task

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. The following sample setup demonstrates one of the common use cases.

You are tasked to enforce authentication requirements on two categories of SP connections:

- For high-value connections, users must authenticate via the X.509 Adapter followed by the PingID Adapter.
- For low-value connections, users can authenticate via the HTML Form Adapter or the X.509 Adapter followed by the PingID Adapter.

You have already created 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.

To fulfill this use case, follow these configuration steps:

Steps

1. Go to the **Identity Provider**# **Selectors** screen.

2. Create an instance of the Session Authentication Selector to account for authentication sessions acceptable for low-value connections.
   a. Click **Create New Instance**.
   b. On the **Type** screen, enter a name (for example, Sessions for low-value connections) and an ID; then select **Session Authentication Selector** from the list.
   c. On the **Authentication Selector** screen, 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

      The following screen capture illustrates the setup.
   d. On the **Summary** screen, click **Done**.
   e. On the **Manage Authentication Selector Instances** screen, click **Save** to keep the newly configured authentication selector instance.
3. Go to the **Identity Provider**# **Policies** screen.
4. Define an authentication policy for high-value connections.
   a. Click **Add Policy**.
   b. Enter a name for the policy; for example, High-value connections.
   c. Under **Policy**, select the instance of the Connect Set Authentication Selector that isolates high-value connections from the rest.
   d. For the **No** policy path, select **Continue**.
   e. For the **Yes** policy path, select the X.509 Adapter instance.
   f. For the **X.509 Adapter instance**# **Fail** policy path, select **Done**.
   g. For the **X.509 Adapter instance**# **Success** policy path, select the PingID Adapter instance.
   h. Click **Options** underneath the PingID Adapter instance and select the X.509 Adapter instance as the source and username as the attribute on the **Incoming User ID** screen.

      Tip: This step is applicable 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 an incoming user ID* on page 344.
   i. For the **X.509 Adapter instance**# **Success**# **PingID Adapter instance**# **Fail** policy path, select **Done**.
   j. For the **X.509 Adapter instance**# **Success**# **PingID Adapter instance**# **Success** policy path, select the authentication policy contract.
   k. Complete the contract mapping for the authentication policy contract.

      The following screen capture illustrates the policy created for high-value connections.
   l. Click **Done**.

5. Define an authentication policy for low-value connections.
   a. Click **Add Policy**.
   b. Enter a name for the policy; for example, Low-value connections.
   c. Under **Policy**, select the instance of the Session Authentication Selector (see *step 2*).
   d. For the **SSO** policy path, select the HTML Form Adapter instance.
   e. For the **HTML Form Adapter instance**# **Fail** policy path, select **Done**.
   f. For the **HTML Form Adapter instance**# **Success** policy path, select the authentication policy contract.
   g. Complete the contract mapping for the authentication policy contract.
   h. For the **Mutual TLS and MFA** policy path, select the X.509 Adapter instance.
   i. For the **X.509 Adapter instance**# **Success** policy path, select the PingID Adapter instance.
   j. Click **Options** underneath the PingID Adapter instance and select the X.509 Adapter instance as the source and username as the attribute on the **Incoming User ID** screen.

      Tip: This step is applicable 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 an incoming user ID* on page 344.

   k. For the **X.509 Adapter instance**# **Success**# **PingID Adapter instance**# **Fail** policy path, select **Done**.
   l. For the **X.509 Adapter instance**# **Success**# **PingID Adapter instance**# **Success** policy path, select the authentication policy contract.
   m. Complete the contract mapping for the authentication policy contract.

      The following screen capture illustrates the policy created for low-value connections.

   n. Click **Done**.
   o. Select the **IdP Authentication Policies** check box to activate authentication polices for IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.

      The following screen capture illustrates the policies created this sample use case.

6. Click **Save** to keep the newly configured authentication policies.

Policies

An authentication policy is a tree of authentication sources, selector instances, or a combination of them, on which 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 are based. Administrators can create one or more authentication policies to fulfill their authentication requirements. As needed, individual policies can be disabled.

Administrators can enable authentication policies on IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows. Administrators can also enable authentication policies on SP-initiated Browser SSO requests received at the `/sp/startSSO.ping` endpoint.

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 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 may 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 if any of the following conditions is met:

- 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 uses cases.

Once inapplicable policies and paths are pruned, 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 may 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 no default authentication source can be found and the **Fail if policy engine finds no authentication source** check box is not selected, 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 both the IdpAdapterId query parameter and the pfidpaid cookie are presented, 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 let the user authenticate through one of them. (If the user selected the **Remember selection** check box and successfully authenticated, PingFederate returns a pfidpaid persistent cookie, identifying the user's preference.)

If the **Fail if policy engine finds no authentication source** check box 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 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.

Defining authentication policies

About this task

You manage authentication policies and settings on the **Authentication Policies** screen.

Steps

1. Select the **IdP Authentication Policies** check box if you want to enable authentication policies for IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.

   This check box is only visible when the IdP role is activated in the **System**# **Protocol Settings**# **Roles & Protocols** screen.

   (This check box is not selected by default.)
2. Select the **SP Authentication Policies** check box if you want to enable authentication policies for SP-initiated Browser SSO requests received by at the `/sp/startSSO.ping` endpoint.

   This check box is only visible when the SP role is activated in the **System**# **Protocol Settings**# **Roles & Protocols** screen.

   (This check box is not selected by default.)

   Note: Selecting the **SP Authentication Policies** check box does *not* enable authentication policies for IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.
3. Select the **Fail if policy engine finds no authentication source** check box if you want PingFederate to deny the requests and to 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 check box is not selected by default.)
4. Click **Add Policy** to create an authentication policy on the **Policy** screen.

   Tip: If you want to create a new policy based on an existing policy, select the **Copy** action.

   a. Enter a name and optionally a description of the policy.
   b. Select an authentication source (an IdP adapter instance or an IdP connection) or a selector instance from the **Policy** list.

      Note: If you start 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 may 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.

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 multifactor authentication 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.

All results (including those based on rules) are displayed under the selected authentication source or selector instance. Each result forms a policy path.

c. For each policy path, select a policy action from the list.

- If additional processing is required, repeat *step 4b*.
- 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 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 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, see *step 5k* in *Enable third-party identity providers* on page 570.

  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 in maintenance releases 9.2.3 and 9.3.3.
- 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 *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.

d. Click **Done** to go back to the **Authentication Policies# Policies** screen.

Your policy is enabled by default. As needed, toggle its status to disable the policy.

5. Optional: Repeat *step 4* 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.

6. If any individual policy is no longer required, select the **Delete** action or toggle its status to disable the policy.
7. Optional: On the **Authentication Policies# Default Authentication Sources** screen, 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.)
8. Optional: On the **Authentication Policies# Tracked HTTP Parameters** screen, 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. For more information, see *Configuring the HTTP Request Parameter Authentication Selector* on page 331.

9. Click **Save**.

Specifying an incoming user ID

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 multifactor 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 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, use the **Options# Incoming User ID** dialog to 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 selecting 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. On the **Authentication Policies** screen, select the applicable authentication policy.
2. On the **Policy** screen, 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.

   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.

   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.

   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. Click **Done** to close the **Incoming User ID** dialog.
6. On the **Policy** screen, continue with the rest of your policy configuration.

Configuring rules in authentication policies

About this task

An authentication source in an authentication policy has two results, **Fail** or **Success**, for which one of the following actions can be set:

- 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 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**.)

PingFederate supports more granular control through the use of rules in authentication policies. 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 such user is a contractor. Your organization mandates that all contractors must authenticate successfully against the OpenToken IdP adapter, followed by another IdP adapter (for example, 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. On the **Authentication Policies** screen, select the applicable authentication policy.
2. On the **Policy** screen, locate the authentication source that you want to define additional successful results for further processing and then click **Rules** underneath it.
3. On the **Rules** dialog select an attribute from the **Attribute Name** list.
4. 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 in the **Condition** list.

   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 that 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.

5. Enter the desired value to be compared against the attribute value from the authentication source in the **Value** field.
6. Enter a unique label in the **Result** field.
7. Optional: Click **Add** and repeat steps *3* to *6* to add another rule.
8. If any individual rule is no longer required, select the **Delete** action.
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.

   By default, this check box is selected. When cleared, the policy engine treats the attempt as a failure when no rules return a match.
10. Click **Done** to close the **Rules** dialog.

    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** screen, continue with the rest of your policy configuration.

Defining authentication policies based on group membership information

About this task

PingFederate provides the capability to configure authentication policies based on group membership information through the use of rules.

Suppose 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 prefer to roll out multifactor 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 multifactor 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)

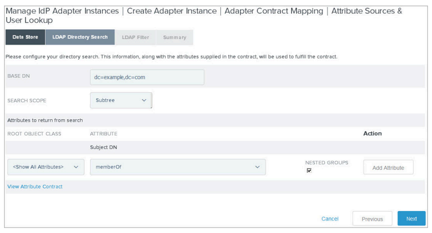
Steps

1. If you have not done so, create a new AD group (for instance, 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 (once 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.
   a. Extend the HTML Form Adapter instance with the memberOf attribute on the **Extended Contract** screen.
   b. Configure PingFederate to fulfill the memberOf attribute from your AD.

      Because the actual groups are nested inside the new group created in *step 1*, configure the IdP adapter contract to pull the memberOf attribute values with nested groups on the LDAP Directory Search screen.

      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

      Manage IdP Adapter Instances | Create Adapter Instance | Adapter Contract Mapping | Attribute Sources & User Lookup
Data Store
LDAP Directory Search
LDAP Filter
Summary
View Attribute Contract
Cancel
Previous
Next

      (For more information, see *Define the IdP adapter contract* on page 486.)
3. On the **Authentication Policies** screen, select the applicable authentication policy.
4. On the **Policy** screen, 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.
    a. Select memberOf from the **Attribute Name** list.
    b. 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.
    c. Enter the DN of the new group created in *step 1* in the **Value** field; for example, `CN=PingIDRequired,OU=IT,DC=example,DC=com`.
    d. Enter a label in the **Result** field; for example, `PingID users`.
    e. 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.

    Your **Rules** dialog should be similar to the following sample:

6. Click **Done** to close the **Rules** dialog.
7. For the new **HTML Form# PingID users** policy path, select your PingID Adapter instance as the policy action from the list.

    Click **Options** (underneath the PingID Adapter instance) to open the **Incoming User ID** dialog. Configure the source of the user ID required by the PingID Adapter, and then click **Done** to close the dialog (see *Specifying an incoming user ID* on page 344).

    Your policy should be similar to the following sample:

    Note that 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.

Your policy should be similar to the following sample:

9. Click **Done** to close the **Policy** screen.
10. On the **Authentication Policies** screen, click **Save**.

Results

Note: Group membership is only one of the possible factors that you can use to define additional policy paths and their policy actions. Generally speaking, you can use any attributes available from the authentication source when configuring rules.

Applying policy contracts or identity profiles to authentication policies

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.

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.

Steps

1. On the **Authentication Policies** screen, select the applicable authentication policy.
2. On the **Policy** screen, 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 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:

This policy has two selector instances (**Test** and **Retail**), two 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.

3. Select **Done** as the policy action for the following paths:

   - **Test**# **No**# **HTML Form**# **Fail**
   - **Test**# **No**# **HTML Form**# **Success**# **Retail**# **Yes**# **PingID**# **Fail**

   At runtime, PingFederate terminates the request and returns an error message to the user.
4. Select the applicable authentication policy contract or local identity profile as the policy action for the rest of the closed-ended paths, namely:

   - **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 may 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.
5. For each selected authentication policy contract (if any), click **Contract Mapping** and then follow the **Manage Authentication Policies**# **Authentication Policy Contract Mapping** wizard to complete the configuration (see *Configuring contract mapping* on page 350).
6. For each selected local identity profile (if any), click **Local Identity Mapping** and then follow the **Manage Authentication Policies**# **Inbound Mapping & Contract Fulfillment** wizard to complete the configuration (see *Configuring local identity mapping* on page 351).
7. Select **Continue** as the policy action for the open-ended path **Test**# **Yes**.

   At runtime, PingFederate skips to the next policy.

   Your policy should be similar to the following sample:
8. Click **Done** to close the **Policy** screen.
9. On the **Authentication Policies** screen, click **Save**.

Configuring contract mapping

Steps

1. Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries.

2. On the **Contract Fulfillment** screen, fulfill the selected contract.

   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* on page 349, 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 IdP connection or IdP adapter instance, you can also use dynamic text, attribute mapping expression (if enabled), and tracked HTTP request parameter (if configured) as the source of fulfillment.
3. Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract (see *Defining issuance criteria for contract or local identity mapping* on page 352).
4. On the **Summary** screen, review your configuration, modify as needed, and then click **Done**.
5. On the **Policy** screen, continue with the rest of your policy configuration.

Configuring local identity mapping

Steps

1. On the **Inbound Mapping** screen, configure the attribute mappings for registration and profile management.

   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.

   Other local identity fields can also be mapped so that PingFederate can streamline the registration process by pre-populating values on the registration page.

   > **Note:** This configuration overrides the default field values configured within the local identity profile (see *Configure a local identity field*).

   This screen does not apply and stays hidden if your use case does not involve registration and profile management (see *Enable third-party identity providers without registration* on page 580).
2. Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries.
3. On the **Contract Fulfillment** screen, 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 under **Source** and the desired local identity field under **Value**.

   > **Note:** If your use case does not involve registration or profile management, the source of fulfillment is limited to the preceding IdP connection or IdP adapter instance, dynamic text, attribute mapping expression (if enabled), and tracked HTTP request parameter (if configured).
4. Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract (see *Defining issuance criteria for contract or local identity mapping* on page 352).
5. On the **Summary** screen, review your configuration, modify as needed, and then click **Done**.

6. On the **Policy** screen, continue with the rest of your policy configuration.

Defining issuance criteria for contract or local identity mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See the following table for more information.

Source	Description
Adapter	Select to evaluate attributes from any preceding IdP adapter instance.
IdP Connection	Select to evaluate attributes from any preceding IdP connection.
Local Identity	Select to evaluate any local identity fields. (Not applicable for the **Contract Mapping** configuration.)
JDBC, LDAP, or other types of data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Tracked HTTP Parameters	Select to evaluate tracked request parameters. Visible and applicable only if at least one HTTP request parameter has been configured on the **Authentication Policies**# **Tracked HTTP Parameters** screen (see).

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   > **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Mapping a policy contract to multiple use cases

About this task

The last step to reuse an authentication policy in multiple SP connections is to map the authentication policy contract into the applicable SP connections.

Generally speaking, for IdP Browser SSO use cases, if you have selected authentication policy contracts in your authentication policies, you must map the authentication policy contracts to the applicable SP connections.

Steps

1. Select the applicable SP connection from the **Identity Provider** menu.
2. On the **Activation & Summary** screen, click **Authentication Source Mapping**.
3. Click **Map New Authentication Policy** and follow the on-screen instructions to map the authentication policy contract into the SP connection.

Results

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 (see *Manage authentication policy contract grant mapping* on page 436).

SP authentication policies

SP authentication policies provide a means for you to impose authentication requirements on SP-initiated Browser SSO requests received at the `/sp/startSSO.ping` endpoint.

When you enabled this optional feature, you are creating policies that the PingFederate SP server can use to find the applicable SP adapter instance to access target applications. It is for this reason that 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 configure one or more entries in the **Service Provider**# **Target URL Mapping** screen to map the TargetResource values to the applicable SP adapter instances.

Note: SP authentication policies are only applicable to SP-initiated Browser SSO requests received at the */sp/startSSO.ping* on page 761 SP application endpoint. They are *not* applicable to unsolicited SSO requests received at the SP protocol endpoints.

It is also worth noting that enabling SP authentication policies does *not* enable authentication policies for IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization code and implicit flows.

For more information and configuration steps, refer to the subsequent sample use cases.

Configure an SP authentication policy for users from one IdP

About this task

You can configure an SP authentication policy to enforce authentication requirements for an IdP connection. Consider the following example.

You are tasked to create an IdP connection to Alpha, which passes two attributes in its assertions, SAML_SUBJECT and samlEmail, on your PingFederate SP server. You are also asked to enforce multifactor authentication 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 SSO profile.

You have already created an SP adapter instance using the **Service Provider**# **Adapters** configuration wizard and completed the last-mile integration with the target application. The SP adapter instance name and ID are **Sample** and `sample`, respectively. On the **System**# **Protocol Settings**# **Federation Info** screen, the base URL for your PingFederate SP server is defined as `https://sso.xray.local:9031`. There are no other IdP connections besides those required to connect with Alpha and Bravo.

This example requires the following components:

- An SP adapter instance deployed, configured, and integrated with the target application.
- An IdP connection to the partner (*step 1*).
- An IdP connection to the third-party IdP that facilitates the multifactor authentication process (*step 2*).
- An authentication policy contract to carry user attributes from the partner to the target application (*step 3*).
- An SP authentication policy (*step 4* and *step 7*).
- An adapter mapping between the authentication policy contract and the applicable SP adapter instance (*step 5*).
- An SP-initiated SSO URL (*step 6*).

To fulfill the requirements:

Steps

1. On the **Service Provider** menu, click **Create New** (under **IdP Connections**).
   a. Follow the connection wizard to create a SAML 2.0 IdP connection to Alpha.

      In this example, Alpha's entity ID is `sso.alpha.local`.
   b. On the **IdP Connection**# **Browser SSO**# **SAML Profiles** screen, make sure that the **SP-Initiated SSO** check box is selected.
   c. On the **IdP Connection**# **Browser SSO**# **User-Session Creation**# **Identity Mapping** screen, 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 in the **IdP Connection**# **Browser SSO**# **User-Session Creation**# **Target Session Mapping** screen. The rest of the steps remain unchanged.

      (This sample configuration uses neither **Account Linking** nor **Account Mapping**.)
   d. 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. On the **Service Provider# Policy Contracts** screen, 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. Generally speaking:
>
> - You map attributes to authentication policy contracts from authentication policies (*step 4* in this example).
> - You map attributes from authentication policy contracts to target applications through adapter mappings (*step 5* in this example).

a. On the **Contract Info** screen, enter a name for this authentication policy contract.

   In this example, the name of the policy contract is `Authenticated`.

b. On the **Contract Attributes** screen, enter `mail` under **Extend the Contract** and click **Add**.

c. Complete the rest of the connection configuration.

When finished, your policy contract has two attributes: subject and mail.

4. On the **Service Provider# Policies** screen, click **Add Policy** to define a policy to enforce the third-party authentication requirement.
   a. On the **Policy** screen, enter a name and a description for this policy, and select **sso.alpha.local (IdP Connection)** as the first policy action from the list.

      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.
   b. For the **sso.alpha.local# Fail** path, select **Done** as the policy action.

      At runtime, PingFederate terminates the request and returns an error message to the user.
   c. For the **sso.alpha.local# Success** path, select **sso.bravo.local** as the policy action.
   d. Click **Options** (underneath **sso.bravo.local**) to relay the user ID (SAML_SUBJECT) from `sso.alpha.local` to `sso.bravo.local`.

      On the **Incoming User ID** dialog, select **IdP Connection (sso.alpha.local)** from the **Source** list and **SAML_SUBJECT** from the **Attribute** list; for example:

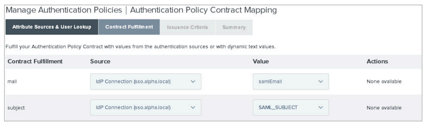
      Click **Done** to close the **Incoming User ID** dialog.
   e. For the **sso.bravo.local# Fail** path, select **Done** as the policy action.

      At runtime, PingFederate terminates the request and returns an error message to the user.
   f. For the **sso.bravo.local# Success** path, select the authentication policy contract created in *step 3*.

      Your policy should be similar to the following sample:
   g. Click **Contract Mapping** (underneath the authentication policy contract) and follow the **Manage Authentication Policies# Authentication Policy Contract Mapping** configuration wizard to configure the fulfillment of the authentication policy contract.

      Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries.

      On the **Contract Fulfillment** screen, select **IdP Connection (sso.alpha.local)** from the **Source** list and the appropriate attributes from the **Value** list to fulfill the authentication policy contract attributes; for example:



Manage Authentication Policies | Authentication Policy Contract Mapping

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.

      Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract.

      On the **Summary** screen, click **Done**.

      On the **Policy** screen, click **Done**.

      On the **Authentication Policies** screen, click **Save**.

5. On the **Service Provider**# **Adapter Mappings** screen, map the authentication policy contract to the SP adapter instance that you have deployed, configured, and integrated with the target application.
   a. Select the authentication policy contract (created in *step 3*) from the **Source Instance** list.
   b. Select the applicable SP adapter instance (**Sample**) from the **Target Instance** list.
   c. Click **Add Mapping**.
   d. Follow the **Authentication Policy Adapter Mappings**# **Mapping Configuration** wizard to create the mapping.

      Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries.

      On the **Adapter Contract Fulfillment** screen, select **Authentication Policy Contract** from the **Source** list and the appropriate contract attributes from the **Value** list to fulfill the SP adapter contract; for example:

      Tip: In this configuration, you are mapping attribute values from the authentication policy contract to the target application through the applicable SP adapter instance.

      Optional: On the **Default Target URL** screen, specify a default target URL for this mapping configuration.

      Optional: On the **Issuance Criteria** screen, 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 the parameter TargetResource (or TARGET) is used and includes its own query parameters, the parameter value must be URL-encoded.

   Any other parameters that contain restricted characters (many SAML URNs, for example) also must be URL-encoded.

   For information about URL encoding, please refer to third party resources, such as HTML URL-encoding Reference (www.w3schools.com/tags/ref_urlencode.asp).
7. When you are ready to test your new use case, go to the **Service Provider**# **Policies** screen, 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**.

Configure SP authentication policies for users from multiple IdPs

About this task

You can configure SP authentication policies to handle different authentication requirements for multiple IdP connections. Consider the following example.

Suppose you have configured the following use cases in an earlier version of PingFederate:

- Two SP adapter instances on the **Service Provider**# **Adapters** screen:

Instance Name	**Instance ID**	**Extended Contract**
Sample	sample	subject and email
Sample Delta	sampleDelta	subject and email

- Three entries on the **Service Provider**# **Target URL Mapping** screen:

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 Mapping** **SP adapter instance name** **(SP adapter instance ID)**
Alpha (sso.alpha.local)	Account Mapping	SAML_SUBJECT and samlEmail	Sample (sample)
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 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%3Fapp%3DAlph%26t%3Daa
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
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

After upgrading to PingFederate 9.3, you have the following new requirements:

- Create new IdP connections to three new partners: Echo, Foxtrot and Golf.
- Enforce multifactor authentication for users from Alpha, Charlie, Echo, and Golf through Bravo. Note that Bravo requires a user ID to be passed in from the original source and returns only the user ID when the users fulfill the multifactor authentication requirement.

The new required components are:

- Two additional SP adapter instances (*step 1*):
  - **Sample Echo** to integrate with Echo's target application.
  - **Sample Golf** to integrate with Golf's target application.
- Four new IdP connections (*step 2*, *step 3*, and *step 4*):

Partner (Federation ID)	Identity Mapping	Attribute Contract	Target Session Mapping SP 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 (*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 (*step 6*).
- Three SP authentication policies to enforce the multifactor authentication requirement (*step 7*, *step 8*, and *step 12*).
- Three adapter mappings between the authentication policy contracts and the applicable SP adapter instances (*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 (*step 10*):
- SSO URLs for all partners (*step 11*).

Follow these steps to fulfill the new requirements:

Steps

1. On the **Service Provider# Adapters** screen, create two new SP adapter instance:

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.
   a. On the **Service Provider** menu, click **Create New** (under **IdP Connections**) and follow the **IdP Connection** wizard to create a SAML 2.0 IdP connection.
   b. On the **IdP Connection# Browser SSO# SAML Profiles** screen, select the **SP-Initiated SSO** check box.
   c. On the **IdP Connection# Browser SSO# User-Session Creation# Identity Mapping** screen, select **No Mapping**.
   d. Complete the rest of the connection configuration.
3. Create IdP connections to Echo and Golf.
   a. On the **Service Provider** menu, click **Create New** (under **IdP Connections**) and follow the **IdP Connection** wizard to create a SAML 2.0 IdP connection to Echo (and then to Golf).
   b. On the **IdP Connection# Browser SSO# SAML Profiles** screen, select the **SP-Initiated SSO** check box.
   c. In the **IdP Connection# Browser SSO# User-Session Creation# Identity Mapping** screen, 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 the **IdP Connection# Browser SSO# User-Session Creation# Target Session Mapping** screen. The rest of the steps remain unchanged.

   This example does not use account linking.

   d. On the **IdP Connection# Browser SSO# User-Session Creation# Attribute Contract** screen, enter `samlEmail` under **Extend the Contract**.
   e. Complete the rest of the connection configuration.
   f. Repeat these steps to create a SAML 2.0 IdP connection to Golf.
4. Create an IdP connection to Foxtrot.
   a. On the **Service Provider** menu, click **Create New** (under **IdP Connections**) and follow the **IdP Connection** wizard to create a SAML 2.0 IdP connection.
   b. On the **IdP Connection# Browser SSO# SAML Profiles** screen, select the **SP-Initiated SSO** check box.
   c. On the **IdP Connection# Browser SSO# User-Session Creation# Identity Mapping** screen, select **Account Mapping**.
   d. On the **IdP Connection# Browser SSO# User-Session Creation# Attribute Contract** screen, enter `samlEmail` under **Extend the Contract**.
   e. In the **IdP Connection# Browser SSO# User-Session Creation# Target Session Mapping** screen, click **Map New Adapter Instance** and follow the **Adapter Mapping & User Lookup** configuration wizard to map the attributes from the assertion to the SP adapter instance **Sample**.

Adapter Contract	Source	Value
email	Assertion	samlEmail
subject	Assertion	SAML_SUBJECT

   f. 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. 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 9* in this example).

a. On to the **Service Provider# Policy Contracts** screen, 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).

b. On the **Contract Info** screen, enter a name for this authentication policy contract.

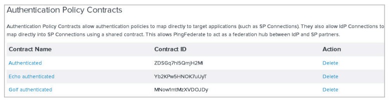
   In this example, the names are `Authenticated`, `Echo authenticated`, and `Golf authenticated`.

c. On the **Contract Attributes** screen, extend the authentication policy contract with an attribute for user's email address; for example, mail.

d. Complete the rest of the connection configuration.

e. Repeat these steps to create an authentication policy contract for users from Echo, and then for users from Golf.

   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	ZDSGq7hI5QmjH2MI	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.

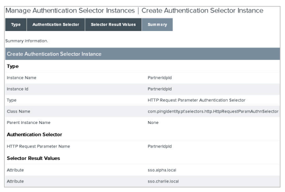
Create an instance of the HTTP Request Parameter Authentication Selector.

a. On the **Service Provider# Selectors** screen, click **Create New Instance**.
b. On the **Type** screen, 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`.
c. On the **Authentication Selector** screen, enter PartnerIdpId in the **HTTP Request Parameter Name** field.
d. On the **Selector Result Values** screen, 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.
e. Complete the rest of the configuration; for example:

   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. On the **Service Provider# Policies** screen, 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 *Configure an SP authentication policy for users from one IdP* on page 354.

a. On the **Policy** screen, enter a name and a description for this policy, and select **sso.echo.local (IdP Connection)** as the first policy action from the list.
b. For the **sso.echo.local# Fail** path, select **Done** as the policy action.
c. For the **sso.echo.local# Success** path, select **sso.bravo.local (IdP Connection)** as the policy action.
d. Click **Options** (underneath **sso.bravo.local (IdP Connection)**) to relay the user ID (SAML_SUBJECT) from `sso.echo.local` to `sso.bravo.local`.
e. For the **sso.bravo.local# Fail** path, select **Done** as the policy action.
f. 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.
g. Repeat these steps to define a new authentication policy to enforce third-party authentication for users from Golf; for example:

   Your policies should be similar to the following samples:

8. On the **Policies** screen, 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*.

   a. On the **Policy** screen, 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.
   b. For the **sso.alpha.local** path, repeat *step 7* to configure the authentication requirement for the `sso.alpha.local` IdP connection.
   c. For the **sso.charlie.local** path, repeat *step 7* to configure the authentication requirement for the `sso.charlie.local` IdP connection.
   d. For the **sso.foxtrot.local** path, repeat *step 7* to configure the authentication requirement for the `sso.foxtrot.local` IdP connection.

      Your policy should be similar to the following sample:

      (Note that this screen capture collapses the **sso.charlie.local** and **sso.foxtrot.local** policy paths for documentation presentation purpose.)

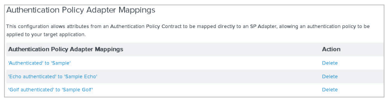
      When you go back to the **Policies** screen, your policies should be similar to the following samples:

   e. Click **Save**.
9. Create adapter mappings.
   a. Go to the **Service Provider# Adapter Mappings** screen.
   b. 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 *Configure an SP authentication policy for users from one IdP* on page 354.

Your mappings should be similar to the following samples:

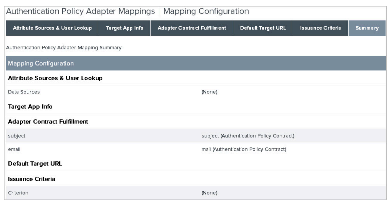


Authentication Policy Adapter Mappings

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 ot the following configuration:


Authentication Policy Adapter Mappings | Mapping Configuration
Attribute Sources & User Lookup
Target App Info
Adapter Contract Fulfillment
Default Target URL
Issuance Criteria
Summary

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:** In essence, you are mapping attribute values from the authentication policy contracts to target applications through the applicable SP adapter instances.

10. Create target URL mappings.

a. Go to the **Service Provider# Target URL Mapping** screen and 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**

Your target URL mappings should be similar to the following samples:

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. **Optional:** Based on the current configuration, you could remove the PartnerIdpId query parameter because it is not required. You could 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. **Optional:** Based on the current configuration, you could 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. **Optional:** Based on the current configuration, you could 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. **Optional:** Based on the current configuration, you could 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 the parameter TargetResource (or TARGET) is used and includes its own query parameters, the parameter value must be URL-encoded.

Any other parameters that contain restricted characters (many SAML URNs, for example) also must be URL-encoded.

For information about URL encoding, please refer to third party resources, such as HTML URL-encoding Reference (www.w3schools.com/tags/ref_urlencode.asp).

12. When you are ready to test your new use case, go to the **Service Provider# Policies** screen, 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**.

Configure SP authentication policies for internal users

About this task

The `/pf/adapter2adapter.ping` endpoint initiates direct IdP-to-SP adapter mapping, mostly intended for the internal users to access resources without maintaining an SP and an IdP connection on the same server.

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 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 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 to PingFederate 9.3, if you want to enforce multifactor authentication for users from Hotel through Bravo, you can create an IdP connection to Bravo and the following authentication policy:

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, thus circumventing 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 (*step 1*).
- An authentication policy contract, **Internal authenticated**, to carry attributes from internal users to the target applications.(*step 2*).
- 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 (*step 3*).
- An authentication policy for the internal users (*step 4*).
- An adapter mapping to map from the authentication policy contracts **Internal authenticated** to the SP adapter instance **Sample Internal** (*step 5*)
- A new SSO URL for the internal users (*step 6*).

Follow these steps to fulfill the new requirements:

Steps

1. On the **Service Provider**# **Adapters** screen, create a new SP adapter instance:

Instance Name	Instance ID	Extended Contract
Sample Internal	sampleInternal	subject and email

2. On to the **Service Provider**# **Policy Contracts** screen, click **Create New Contract** to create an authentication policy contract.

   In this example, the name of the policy contract is `Internal authenticated`.

   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. Generally speaking:

   - You map attributes to authentication policy contracts from authentication policies (*step 4* in this example).
   - You map attributes from authentication policy contracts to target applications through adapter mappings (*step 5* in this example).

3. On the **Service Provider**# **Selectors** screen, 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.

   In this example, the name and ID are both `Internal users`.

   Note: 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. Another approach is to 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.

4. Go to the **Service Provider**# **Policies** screen to configure your policies as follows:

   MY TITLE With PingID Adapter

   MY TITLE With CIDR Authentication Selector

5. On the **Service Provider**# **Adapter Mappings** screen, create a mapping from the new authentication policy contract (`Internal authenticated.`) to the new SP adapter instance (`sampleInternal`).

   On the **Default Target URL** screen, enter `https://sso.xray.local:9031/SpSample/MainPage/?app=Internal&t=i`.

6. Update the SSO URL for your internal users to:

   https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleInternal

Results

Policy contracts

Authentication policy contracts, formerly known as connection mapping contracts, provide PingFederate administrators the following benefits:

- The capability to build an attribute contract with attribute values from multiple authentication sources or data store queries through an authentication policy.

- The flexibility to map only the policy contract to a connection. Authentication sources in the policy leading up to the contract are not required to be mapped into the connection. 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 SP connections, 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 (see *Federation hub use cases* on page 127).

Manage policy contracts

About this task

You manage authentication policy contracts (formerly known as connection mapping contracts) in the **Authentication Policies**# **Policy Contracts** screen.

Steps

- Click **Create New Contract** to create a new authentication policy contract.
- Edit an existing authentication policy contract by clicking its name.
- Check its usage.
- Use the **Delete** or **Undelete** workflow to remove an authentication policy contract (if the authentication policy contract is not in use) or cancel the removal request.

Edit contract information

About this task

You can enter or modify the contract name in the **Contract Info** screen. When complete, click **Done**.

Define contract attributes

About this task

You manage the user attributes in the authentication policy contract in the **Contract Attributes** screen. Every authentication policy contract comes with a subject attribute. You extend the contract with additional attributes, as needed.

Steps

- Enter an attribute under the **Extended the Contract** column, and then click **Add**.

  Attribute names are case-sensitive and must suit the needs of your partners. Repeat to add more attributes as needed.
- Modify an existing attribute or undo changes made by using the **Edit**, **Update**, or **Cancel** workflow.
- Click **Delete** to remove an existing attribute.

Review the policy contract

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Adapter Mappings

This configuration allows attributes from an authentication policy contract to be mapped directly to an SP adapter instance, allowing 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.

Configure authentication policy adapter mappings

Steps

1. Go to **Service Provider**# **Adapter Mappings** screen.
2. Select the applicable authentication policy contract from the **Source Instance** list.
3. Select the SP adapter instance integrated with your target application from the **Target Instance** list.
4. Click **Add Mapping**.
5. Follow the **Authentication Policy Adapter Mappings**# **Mapping Configuration** wizard to create the mapping.
    a. Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure one ore more data store queries to fulfill the SP adapter contract.

       Queries are executed in the order as shown on the **Attribute Sources & User Lookup** screen. Use the up and down arrows as needed to adjust the order.

       If a required attribute (such as the user identifier of an adapter) cannot be fulfilled, the request fails.

       For more information, see *Fulfillment by data store queries* on page 870.
    b. On the **Adapter Contract Fulfillment** screen, select a source and an attribute to fulfill the SP adapter contract.

       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 data store queries, dynamic texts, or results from OGNL expressions.
    c. Optional: On the **Default Target URL** screen, specify a default target URL for this mapping configuration.
    d. Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an SP adapter contract (see *Define issuance criteria for adapter mapping*).
    e. On the **Summary** screen, review the configuration and modify as needed. When complete, click **Done**.
6. On the **Authentication Policy Adapter Mappings** screen, click **Save**.

Define issuance criteria for adapter mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See 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:** 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   ⓘ **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      ⓘ **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Sessions

Application sessions

Application sessions apply to PingFederate applications hosted on its user-facing endpoints; for example, 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 (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 SSO or PingFederate applications.

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 data store for persistent authentication sessions* on page 182.

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 may or may 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 SLO application endpoints to terminate adapter sessions, add sessions to the session revocation list as users sign off, or do both. Publishing revoked sessions is recommended in order to provide a secure single logout (SLO) experience with PingAccess® deployments.

Configure tracking options for logout

Steps

1. Go to the **Sessions** screen from the **Identity Provider** or **Service Provider** menu.
2. Optional: Select the **Track Adapter Sessions for Logout** check box to enable SLO for all adapter instances on a per-user basis.

   When selected, if a user triggers a web application to send a logout request to one of the PingFederate SLO application endpoints (`/idp/startSLO.ping` and `/sp/startSLO.ping`), the user is logged out locally, including adapter sessions established through **Adapter-to-Adapter Mappings** configuration. If the applicable connections support SLO (for instance, a WS-Federation connection or an SLO-enabled SAML 2.0 connection), logout requests are sent to the partners based on the protocol specifications.

This check box is not selected by default.

3. Optional: Select the **Track Revoked Sessions on Logout** check box to add the associated sessions to the revocation list on logout.

   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`).

   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. (For more information, see *Manage session validation settings* on page 451).

   This check box is selected by default for new installations.

4. Optional: Modify the default value of the **Session Revocation Lifetime** field (in minutes).

   This field defines the amount of time in minutes until the revoked sessions are removed from the revocation list for optimal performance.

   You may 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** to keep your configuration.

Configure application sessions

Steps

1. Go to the **Sessions** screen from the **Identity Provider** or **Service Provider** menu.
2. Optional: 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 may enter an integer between `1` and `1576800`, representing a range of one minute to 1,095 days. You may 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 may enter an integer between `1` and `1576800`, representing a range of one minute to 1,095 days. You may 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. Click **Save** to keep your configuration.

Configure authentication sessions

Steps

1. Go to the **Sessions** screen from the **Identity Provider** or **Service Provider** menu.
2. Optional: Configure the global policy and timeout settings under **Authentication Sessions**.
   a. 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.

   b. Select the **Make Authentication Sessions Persistent** check box if your use cases require a longer session duration or a greater resilience against restarts of PingFederate and browsers.

      This check box is not selected by default.

      Note: Persistent authentication sessions require an external storage.

   c. 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 may enter an integer that represents a time period between 1 minute and 1,095 days. You may also empty the value to indicate that the inactivity timeout value should match the maximum lifetime. The default inactivity timeout value is `60` minutes.

Field	Description
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 may enter an integer that represents a time period between 1 minute and 1,095 days. You may 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 inactivity timeout value is `480` minutes (eight hours).

3. Optional: Configure policy and settings for individual authentication sources under **Overrides**.

a. Select an IdP adapter instance or an IdP connection from the **Authentication Source** list.

b. Configure individual policy for the selected authentication source as follows:

Global policy (under Authentication 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.

The **Enable Sessions** check box is not selected by default.

Note: Keep in mind that 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.

c. 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.

d. 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 may enter an integer that represents a time period between 1 minute and 1,095 days. You may 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 may enter an integer that represents a time period between 1 minute and 1,095 days. You may 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**.

e. 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.

f. Click **Add**.

g. Repeat these steps to configure individual policy and settings for additional authentication sources.

   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.

4. Click **Save** to keep your configuration.

Results

When PingFederate authentication sessions are enabled, you may configure session-validation options for your OAuth use cases. Such optional settings enable you to conjoin the validity of access tokens and the authentication sessions of the users. For more information, see *Manage session validation settings* on page 451.

OAuth configuration

To use the OAuth authorization server (AS), start by enabling that role for PingFederate under **System# Protocol Settings# Roles & Protocols** screen. Then configure the OAuth AS settings, as described in this section.

ⓘ **Tip:** Service providers may also add OAuth capabilities to the Browser SSO configuration for IdP connection partners, see *Configure OAuth attribute mapping*.

Configure OAuth use cases

About this task

Use the **OAuth Server** menu to configure various settings to support the grant types that your applications require.

Steps

1. Configure the authorization server (AS) settings in the **OAuth Server# Authorization Server Settings** screen.
2. Define any number of optional common scopes and exclusive scopes, create scope groups from any optional scopes as needed, and enter an optional description for the default scope using the **OAuth Server# Scope Management** configuration wizard.
3. Create one or more access token management instances using the **OAuth Server# Access Token Management** configuration wizard.

   This is where you define the access token attribute contract for any given access token management instance.
4. Configure one or more entries to map attributes from authentication sources to the persistent grants.

   Authorization Code or Implicit

   - Map attributes from an IdP adapter instance to the persistent grants using the **OAuth Server# IdP Adapter Mapping** configuration wizard.
   - Map attributes from an IdP connection to the persistent grants using the **IdP Connection# Browser SSO# OAuth Attribute Mapping** configuration wizard.
   - Create an authentication policy contract (APC) using the **Policy Contracts** configuration wizard, 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 **OAuth Server# Authentication Policy Contract Mapping** configuration wizard.

     ⓘ **Tip:** If you are using a combination of authentication policies, APCs, and APC mappings, you can skip the **IdP Adapter Mapping** and **OAuth Attribute Mapping** configurations.

   Resource Owner Password Credentials

   - Map attributes from a password credential validator instance to the persistent grants using the **OAuth Server# Resource Owner Credential Mapping** configuration wizard.

   Note that 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 using the **OAuth Server**# **Access Token Mapping** configuration wizard. Additionally, you can configure a mapping for clients using the client credential grant type.

   Note that 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* on page 109.)
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 JWT Bearer or SAML 2.0 Bear assertion grants flow, configure a mapping on the **IdP Connection**# **OAuth Assertion Grant Attribute Mapping** screen.

   Note that 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 **OAuth Server**# **OpenID Connect Policy Management** configuration wizard if you support OpenID Connect use cases *and* the **OpenID Connect** protocol is enabled in **System**# **Protocol Settings**# **Roles & Protocols** screen.
9. Create one or more OAuth clients in the **OAuth Server**# **Client Management** screen.
10. Optional: Configure client settings and registration policies for dynamic client registration.
11. Optional: Configure client session management settings.

Enable the OAuth AS role

About this task

To configure OAuth use cases, you must enable the authorization server (AS) role in PingFederate.

ⓘ **Tip:** If your PingFederate license does not include the OAuth AS capabilities, please contact sales@pingidentity.com.

Steps

1. Go to the **System**# **Protocol Settings**# **Roles & Protocols** screen.
2. Select the **Enable OAuth 2.0 Authorization Server (AS) role** check box.
3. Optional: Select the **OpenID Connect** check box if you intend to support OpenID Connect use cases as well.

Configure AS settings

About this task

The **Authorization Server Settings** screen provides overall controls over the usage and behavior of PingFederate as an OAuth authorization server (AS), including the policies and settings for various grant types, refresh-tokens, persistent grants, and ID tokens.

Steps

1. Go to the **OAuth Server**# **Authorization Server Settings** screen.

2. Configure AS settings that suit your use cases.

Refer to the following table for detailed information about 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`.
Track User Sessions for Logout	When selected, PingFederate links the sessions for IdP adapters that are used by clients to the PingFederate authentication session of the user (the resource owner). When a user initiates logout, PingFederate sends (via the browser) logout requests to close the adapter sessions. If the **OpenID Connect** protocol is enabled in **System**# **Protocol Settings**# **Roles & Protocols** screen, selecting this check box also allows per-client logout endpoints to be defined, which will be invoked during logout. The check box is not selected by default.
Refresh Token and Persistent Grant Settings	
Persistent Grant Max Lifetime	This field controls whether persistent grants should expire, and if so, the default maximum lifetime for persistent grants. ▪ Select **Grants Do Not Expire** and 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 a unit of measurement (minutes, hours, or days) from the list. The default selection is **Grants Do Not Expire**. **Note:** You may override the expiration in individual client records or grant-mapping configurations. The latter takes precedence and requires an extended persistent grant attribute PERSISTENT_GRANT_LIFETIME.

Field	Description
Persistent Grant Idle Timeout	This field controls whether persistent grants should expire due to inactivity (regardless of whether the maximum lifetime has been reached), and if so, 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 (minutes, hours, or days) from the list. If an idle timeout value is configured, the idle timeout window slides when a persistent grant is updated (see). When an idle timeout value is configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity, or are revoked or removed. When an idle timeout value is configured with a maximum lifetime, persistent grants remain valid until they expire (due to inactivity or lifetime expiration) or are removed from the grant storage. 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 may override the expiration in individual client records.
Refresh Token Length (Characters)	The length of the refresh tokens in number 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. **Note:** New refresh tokens are not issued during the interval defined by the **Minimum Interval to Roll Refresh Tokens** field. When not selected, each refresh token is used until it becomes invalid (either by manually revoking or by some other security setting that renders it invalid). This check box is not selected by default.
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
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 type (or grant types) that you want PingFederate to reuse the existing grant 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 one (the most recently issued) is valid; 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** (selected by default) ▪ **Authorization Code** ▪ **Resource Owner Password Credentials** When the **Implicit** check box 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** check box is selected, the same is true *if* the **Bypass Authorization for Previously Approved Persistent Grants** check box 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. The check box is not selected by default.

Allow Unidentified Clients to Request Extension Grants	When selected, PingFederate allows user-initiated or client-initiated events (for example, 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:
	▪ 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`
	
	The check box is not selected by default.

Token Endpoint Base URL	When clients authenticate via 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 take into consideration as well when validating the aud parameter. If configured, the OpenID Provider 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.

Field	Description
Persistent Grant Extended Attributes	
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 Mapping** ▪ **OAuth Attribute Mapping** ▪ **Authentication Policy Contract Mapping** ▪ **Resource Owner Credentials Mapping** This field has no default entry. Multiple entries are allowed.
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 behavior applies only when using the authorization code grant type *and* when the **Reuse Existing Persistent Access Grants for Grant Types** check box is selected. The check box is not selected by default. **Note:** You may override this setting in individual client records.

Field	Description
Consent User Interface	This field controls whether PingFederate or a trusted web application should handle consent approval. **Default** Select **Default** and let PingFederate handles 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. 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 (version 1.5 or a subsequent version), you may select it from the list. The expectation is that the trusted web application is integrated with PingFederate through this Reference ID Adapter instance. Your development team may also create a custom adapter using the PingFederate SDK. For more information, refer to the Javadoc for 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 located under the `<pf_install>/pingfederate/sdk` directory. Once deployed, you may create an instance of the custom adapter and select it from the list. **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** from the list.

OAuth Administrative Web Services Settings

Field	**Description**
Password Credential Validator	If you want to manage clients using the OAuth Client Management Service or to manage persistent grants using the OAuth Access Grant Management Service, select a Password Credential Validator instance from the list. This setting has no default selection. When no validator is selected, neither services can be used.

Persistent Grant Management API

The Persistent Grant Management 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. (For more information, see *OAuth Persistent Grant Management API* on page 848.)

Access Token Manager	Select an Access Token Management instance under which the access tokens issued can be used by one or more clients to manage persistent grants on their users' behalves. Note that such clients must also be configured to use this Access Token Management instance in their client configurations.
Required Scope	Select the scope that PingFederate looks for in access tokens prior to granting clients the permission to manage persistent grants on their users' behalves. Note that 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`

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 JSON Web Key Sets. For more information about CORS, please refer to *W3C's recommendation on Cross-Origin Resource Sharing* (www.w3.org/TR/cors).

Sample entry	Expected behavior
`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: While 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. Multiple entries are allowed.

Device Authorization Grant Settings

The *OAuth 2.0 Device Authorization Grant* (tools.ietf.org/html/draft-ietf-oauth-device-flow) specification defines the process that allows a user to grant authorization to a device using a browser on a second device, such as a smartphone or a computer. For more information, see *Device authorization grant* on page 52.

Field	**Description**
User Authorization URL (Optional)	This field controls whether PingFederate should use a different URL, perhaps for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses (see *Device authorization endpoint* on page 808). 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 (see *User authorization endpoint* on page 811). Moreover, it 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 may override this setting in individual client records.

Field	Description
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 (see *Device authorization endpoint* on page 808). 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 (see *User authorization endpoint* on page 811). 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.
Pending Authorization Timeout (seconds)	The lifetime of an activation code (the user_code parameter value) in seconds. The default value is `600`. **Note:** You may 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 may override this setting in individual client records.
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 controls whether PingFederate should skip this confirmation step. Select the **Bypass Activation Code Confirmation** check box if you want PingFederate to skip the confirmation step. This check box is not selected by default. **Note:** You may override this setting in individual client records.

External consent user interface

As use cases evolves towards giving users more control over their data, it is becoming more important to provide detailed information about the requests. While scope descriptions may 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 account (or accounts) to be made available to the client.

To use an external web application for consent approval, configure the **Consent User Interface** setting on the **OAuth Server# Authorization Server Settings** screen. (Choose the **External** option and then configure the **External Consent IdP Adapter** and **External Consent Scopes Attribute** settings accordingly.)

Responsibilities of the external web application

From PingFederate's point of view, delegating consent approval to an external web application means that this is a web application that PingFederate can trust. PingFederate expects this trusted web application to take on the following responsibilities:

- Retrieve from PingFederate the list of requested scopes 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 the user interface elements to approve or deny the requested scopes.
- Validate that the approved scopes found in the response from the resource owner do not exceed the requested scopes.

  > **Important:** This validation guards against unauthorized access in the event that the response is tampered and the original approved scopes are compromised.
- As needed, modify the approved scopes prior to returning them to PingFederate.

  This flexibility allows the web application to override authorization decisions by modifying the approved scopes before returning them to PingFederate.
- Return to PingFederate the list of approved scopes in a secure manner.

Handling of approved scopes

Default consent user interface

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 do not exceed the requested scopes. If this validation passes, PingFederate adds the approved scopes to the access token; otherwise, PingFederate returns an error (`invalid_scope`) to the client.

External consent user interface

When the option to use an external consent user interface is chosen, PingFederate delegates consent approval to an external web application. Because PingFederate trusts this web application, it always adds the scopes returned by the trusted web application to the access token, regardless of whether the returned scopes have already been defined in the system. That being said, 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 mapping* on page 427 and *Token mapping* on page 445.

Scopes and scope management

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 users (the resource owner) approves, the authorization server issues an access token with such scopes.

Static scopes versus dynamic scopes

As an authorization server, PingFederate supports the concepts of static scopes and dynamic scopes. A static scope is defined by using a text value; for example, `read_bank_account`. A dynamic scope is defined by using a text value with a variable component represented by a wildcard; for example, `read_bank_account_txn:*`. As illustrated, 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 instance, when a client sends an authorization request (or a token request if authorization is not required) with a requested scope of `read_bank_account_txn:1234`, because PingFederate is able to match the requested scope to the dynamic scope pattern of `read_bank_account_txn:*`, PingFederate can issue an access token with the requested scope (`read_bank_account_txn:1234`).

Scope groups

For ease of management and subsequent client interactions, PingFederate provides the capability to create multiple groups of static scopes. Essentially, a scope group is a *super scope* that a client can reference in applicable OAuth 2.0 protocol interactions. Once authorized, clients may 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; multiple sub scopes are allowed. As needed, 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

When a client submits an authorization request, it can optionally include one or more scope values. If the request is authorized, PingFederate issues an access token to the client. As the client brings the access token to a resource server (RS) to access protected resources, the RS may contact PingFederate to validate the access tokens. Scope groups are not expanded in JWT-based access tokens or token introspection responses by default. As needed, 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** page always presents the description of the requested scope groups (if any).

Common scopes versus exclusive scopes

Furthermore, PingFederate provides the flexibility to manage scopes and scope groups in two buckets: *common* versus *exclusive*.

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 **OAuth Server**# **Client Settings**# **Scope Constraints** configuration. Note that the **Scope Constraints** configuration is shared across all clients registered via dynamic client registration. If a certain client requires a different set of common scopes or scope groups, you can 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 via the Dynamic Client Registration protocol can also be configured to allow a subset of exclusive scopes or scope groups based on the **OAuth Server**# **Client Settings**# **Scope Constraints** configuration. Note that **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 a scope group) is either a common scope (or a common scope group) or an exclusive scope (or an exclusive group). Duplicate scopes and scope groups are not allowed. In addition, scope and scope group values are case-sensitive.

Tip: For scopes that are intended for the majority of clients, create them as common scopes. For scopes that should be limited to the minority of clients, create them 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 controls whether all existing common scopes and scope groups (and those created in the future) or only the select few should be made available to the client.

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; that is, 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 common scopes and scope groups and those created in the future are available to the client. This is the default behavior.

Exclusive Scopes

This setting 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; that is, 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 detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section, found later in this topic.

Dynamic scopes

A dynamic scope is defined by using a text value with a variable component represented by a wildcard, the asterisk character (`*`). 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`

Only one variable component is permitted. Additionally, backslashes (`\`) and double quotation marks (`"`) are not allowed in neither the prefix nor 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 request (or a token request if authorization is not required) 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** page with the description associated with the matched static scope or scope group. At the end, 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 a partial match to one or more dynamic scopes, the partial match with the highest number of matched characters in the prefix, the 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. Note that if PingFederate should issue an access token, the token is issued with the requested scope, not the matched dynamic scope pattern.

For example, suppose you added the following dynamic scopes:

Common Scopes	Exclusive Scopes
`*123`	`zSomeExclusiveScope`
`*12345`	

Common Scopes	Exclusive Scopes
a*c#123	
ab*#123	
xy*123	
xy*	

You also added a client without any common scope restrictions. (In other words, 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. For instance, 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 follows:

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 whether 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 controls whether all existing common scopes and scope groups (and those created in the future) or only the select few should be made available to the client. You can use this setting to restrict certain common dynamic scopes so that they become invalid for a client.

The **Exclusive Scopes** setting controls whether any exclusive scopes and scope groups should be made 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, for that matter) as candidates when it tries 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 become eligible candidates. If PingFederate matches a requested scope to an exclusive dynamic scope, where such scope is not chosen to be available to the client, PingFederate returns an `invalid_scope` error message to the client. This remains true even if a lesser partial match to an available common dynamic scope.

For instance, suppose you updated your previous sample scope configuration as follows:

Common Scopes	Exclusive Scopes
`*123`	`xy*123`
`*12345`	`zSomeExclusiveScope`
`a*c#123`	
`ab*#123`	
`xy*`	

The following table illustrates the expected results when the client sends an authorization request with a requested scope of `xy#123` based on various client configuration scenarios.

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`. Note that the exclusive dynamic scope of `xy*123` is not taken into consideration because the **Exclusive Scopes** check box is not selected, a per-client configuration scenario PingFederate does not consider any exclusive scopes and scope groups as eligible candidates.	`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.

Per-client scope management settings	Result	Variable component from the requested scope
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. Note that the exclusive dynamic scope of `xy*123` is not taken into consideration because the **Exclusive Scopes** check box is not selected, a per-client configuration scenario PingFederate does not consider any exclusive scopes and scope groups as eligible candidates.	Not applicable.
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, you enter a value and a description for the scope or the scope group. The 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 control the text appears on the **Request for Approval** page.

Default scope

The default scope is the permissions implied 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 (for example, `oauth.approval.page.template.defaultScope`); 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.

Static scopes and scope groups

Similar to the default scope, you may enter simple descriptions or localize the descriptions by using the PingFederate localization framework.

Dynamic scopes

You may enter simple descriptions. You may also use a mix of text and scope-description variables. ${scope} represents the requested scope, whereas ${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 show them as is on the **Request for Approval** page.

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 approach opens up the opportunity 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 may direct the developers to your PingFederate's 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. As needed, you can optionally customize the response to include such scopes and scope groups.

Define scopes

About this task

PingFederate provides the flexibility to manage scopes and scope groups in two buckets, common versus exclusive. Common scopes and scope groups are optional. If defined, they are available to all clients. As needed, 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. If defined, 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. Furthermore, you have the options to create static scopes, static scope groups, and dynamic scopes. Scope groups allows clients to request a super scope and optionally downgrade to a subset of it at a later time. 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* on page 390.

You manage scopes, scope groups, and the default scope description using the **OAuth Server# Scope Management** configuration wizard. Configuration steps for common and exclusive scopes (and scope groups) are identical, with the exception of the configuration screens, which are **Common Scopes** and **Exclusive Scopes**.

Note: A scope (or a scope group) is either a common scope (or a common scope group) or an exclusive scope (or an exclusive group). Duplicate scopes and scope groups are not allowed.

Tip: For scopes that are intended for the majority of clients, create them as common scopes. For scopes that should be limited to the minority of clients, create them as exclusive scopes. As needed, you may organize common (or exclusive) static scopes into common (or exclusive) scope groups.

Steps

1. Go to the **OAuth Server# Scope Management** screen.
2. On the **Common Scopes** screen (or the **Exclusive Scopes** screen), configure any number of common (or exclusive) static scopes.
   a. Enter a static scope value and its description under **Scope Value** and **Scope Description**.

      Scope Value

      Scope value is case-sensitive. A requested scope value of `email` does not match a configured static scope value of `Email`.

      Note: Do not use backslashes (`\`) or double quotation marks (`"`) in the **Scope Value** field.

      Scope Description

      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 example, `oauth.approval.page.template.someScope`); 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.

   b. Click **Add**.
   c. Repeat to define additional static scopes.

      Display order does not matter.

3. On the **Common Scopes** screen (or the **Exclusive Scopes** screen), configure any number of common (or exclusive) dynamic scopes.

a. Enter a dynamic scope pattern and its description under **Scope Value** and **Scope Description**.

Scope Value

You must use a case-sensitive text value with a wildcard, the asterisk character (`*`). 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`

> **Note:** Only one variable component is permitted. Additionally, backslashes (`\`) and double quotation marks (`"`) are not allowed in neither the prefix nor the suffix.

Scope Description

You may enter a simple description. You may also use a mix of text and scope-description variables. ${scope} represents the requested scope, whereas ${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 show them as is on the **Request for Approval** page.

In general, 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.

b. Select the check box under **Dynamic**.
c. Click **Add**.
d. Repeat to define additional dynamic scopes.

Display order does not matter.

4. On the **Common Scopes** screen (or the **Exclusive Scopes** screen), configure any number of common (or exclusive) scope groups.

a. Enter a scope group value and its description under **Scope Group Value** and **Scope Group Description**.

Scope Group Value

Generally speaking. 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 as well.

> ⓘ **Note:** Do not use backslashes (`\`) or double quotation marks (`"`) in the **Scope Group Value** field.

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 example, `oauth.approval.page.template.someScopeGroup`); 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.

b. Select at least one static scope under **Sub Scopes**.

   The administrative console filters out dynamic scopes because the scope-grouping capability is reserved for static scopes only.

c. Click **Add**.

d. Repeat to define additional scope groups.

   Display order does not matter.

5. Optional: On the **Common Scopes** screen (or the **Exclusive Scopes** screen), 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.

> ⓘ 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. In addition, errors can 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** screen, enter a description for the default scope.

The default scope is the permissions implied when no scope and scope group values are indicated or in addition to any scope or scope group values.

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 example, `oauth.approval.page.template.defaultScope`); 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.

7. Click **Save** to keep your configuration.

Results

Scopes and scope groups represent access to resources or APIs on the resource server (RS). For clients supporting the OpenID Connect protocol, you may direct the developers to your PingFederate's 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. As needed, you can optionally customize the response to include such individual scopes and scope groups.

Configure client settings

About this task

Use the **Client Settings** configuration wizard to configure dynamic client registration settings.

Steps

1. Select any applicable screen and modify settings, as needed.
2. Click **Next** to move on to the next screen or click **Save** to retain your changes.

Configure dynamic client registration settings

About this task

Dynamic client registration allows developers to register OAuth clients via an API based on open standards. PingFederate supports various client metadata (see *Supported client metadata* on page 400). If specific use cases require additional metadata, add them as extended properties on the **System# Extended Properties** screen.

Important: Because dynamic client registration can expose your server to unwanted client registrations, it is recommended to protect 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 data store, such as a database or LDAP directory. If you have not yet switched from on-disk client storage (default) to an external data store, refer to *Defining an OAuth client data store* on page 191 for instructions to complete the task.

You may continue with the rest of the configuration; however, dynamic client registration remains inactive until an external client storage is defined.

Steps

1. Go to the **OAuth Server# Client Settings# Dynamic Client Registration** screen.
2. If you want to enable dynamic client registration, select the relevant check box.

   (This check box is not selected by default.)
3. Optional: Select the check box to mandate the requirement of an initial access token

   Important: Although optional, it is recommend to select this option to add a layer of protection against unwanted client registrations.

   If selected, you must also select the required scope (or scope group) from the list.

   Furthermore, developers must be set up to obtain access tokens with the required scope (or scope group) from your PingFederate AS server. For example, you may 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 may also write a custom web app that does the OAuth flow to obtain access tokens on behalf of the developers as they make their requests.

Results

When dynamic client registration is active, developers can send client registrations to the `/as/clients.oauth2` endpoint to create OAuth clients dynamically.

Supported client metadata

Metadata field	Metadata description
client_name	A descriptive name for the client instance. This name appears when the user is prompted for authorization.

Metadata field	Metadata description
token_endpoint_auth_method	The client authentication method. Allowed values: ▪ `none` ▪ `client_secret_basic` ▪ `client_secret_post` ▪ `tls_client_auth` For more information, see *Mutual TLS Profiles for OAuth clients* (tools.ietf.org/html/draft-ietf-oauth-mtls-01). ▪ `private_key_jwt` For more information, see *Client Authentication* in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication).
tls_client_auth_subject_dn	The subject DN of the client certificate. Required if `tls_client_auth` is the value of the token_endpoint_auth_method parameter
token_endpoint_auth_signing_alg	The signing algorithm that the client must use to sign the JWTs for client authentication. Applicable only when the token_endpoint_auth_method parameter is provided with a value of `private_key_jwt`. Allowed 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 either a 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 and , respectively.) 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 may send its authorization requests using request objects. Allowed 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 either a 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 and , respectively.) When this parameter is not provided, the client can use any of the supported signing algorithms. For more information about request object, please refer to the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#RequestObject).
jwks_uri, and jwks	The URL of the JSON Web Key Set (JWKS) or the actual JWKS from the client. Only one of them is required if the client is configured to use the private_key_jwt client authentication method, to transmit request parameters in signed request objects, or to transmit CIBA request parameter in signed request objects (or to do any of them) so that PingFederate can verify the authenticity of the JWTs. In addition, either may also 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 runtime processing, see .) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see .) Finally, if the client is configured to encrypt ID tokens using an asymmetric encryption algorithm (see the **ID Token Key Management Encryption Algorithm** setting), either the JWKS URL or the actual JWKS must be provided.

Metadata field	Metadata description
redirect_uris	An array of one or more redirection URIs to which the OAuth AS may redirect the resource owner's user agent after authorization is obtained. At least one redirection URI is required by the authorization code and implicit grant types.
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, for which a client can request.
grant_types	An array of one or more grant types, for which a client can request. Allowed 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) ▪ `urn:pingidentity.com:oauth2:grant_type:validate_ bearer` (For more information about each grant type, see *Grant types*.)

Metadata field	Metadata description
response_types	An array of one or more response types, for which a client can request. Allowed values: ▪ `code` ▪ `code id_token` ▪ `code id_token token` ▪ `code token` ▪ `id_token` ▪ `id_token token` ▪ `token` For more information about these response types, see Definitions of Multiple-Valued Response Type Combinations (openid.net/specs/oauth-v2-multiple-response-types-1_0.html#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. Additionally, it is worth noting that the response types and grant types 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.

response type	grant types
`code`	`authorization_code`
`code id_token`	`authorization_code` and `implicit`
`code id_token token`	`authorization_code` and `implicit`
`code token`	`authorization_code` and `implicit`
`id_token`	`implicit`
`id_token token`	`implicit`
`token`	`implicit`

Metadata field	Metadata description
id_token_signed_response_alg	The JSON Web Signature (JWS) algorithm required for the OpenID Connect ID 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 either a 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 and , 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.
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

Metadata field	Metadata description
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: ▪ `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
backchannel_token_delivery_mode	The token delivery method supported by the client. 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 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. Allowed 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 either a 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 and , 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 that when user code support is enabled, the associated CIBA request policy must also be user code-enabled.
sector_identifier_uri	
subject_type	

Configure scope constraints

About this task

On the **Scope Constraints** screen, optionally configure which scopes or scope groups that developers can request when registering clients using dynamic client registration.

Note: This configuration is shared among all clients created through dynamic client registration. If a certain client requires a different set of common scopes or 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. In addition, scopes can also be overridden by client registration policies enforced during dynamic client registration.

Steps

1. Go to the **OAuth Server**# **Client Settings**# **Scope Constraints** screen.
2. If you want to restrict clients created via 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.

   Note that 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. Providing 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. If they fail to do so, clients resulting from otherwise valid requests are also configured without any common scopes or scope groups.

3. If you want to allow clients created via the Dynamic Client Registration protocol to request for a subset of exclusive scopes, select the one or more applicable exclusive scopes in the **Allowed Exclusive Scopes** setting.

   Note that 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.

Results

Restricting common scopes and allowing exclusive scopes are not mutually exclusive. You can configure both options based on your use cases.

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, however, is always allowed for and available to all clients. For detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section in *Scopes and scope management* on page 390.

If you configure both options, developers must send client registrations with the desired common and exclusive scopes.

Manage client configuration defaults

About this task

On the **Client Configuration Defaults** screen, specify the default settings that are proprietary to PingFederate for clients created via the OAuth 2.0 Dynamic Client Registration protocol.

While 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. Alternatively, you may 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 the **OAuth Server# Client Settings# Client Configuration Defaults** screen.
2. Optional: Modify the default values as needed.

   Refer to the following table for detailed information about each field.

Field	Description
Private Key JWT - Replay Prevention	Determines whether PingFederate mandates a unique signed JWT from the client for each request when the client is configured to authenticate via the private_key_jwt client authentication method, to transmit request parameters using in 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 (see).

Field	Description
Require Signed Request	Indicates 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 OpenID Connect specification calls this JWT a *request object*. This check box is not selected by default.
Default Access Token Manager	The default Access Token Management (ATM) instance for this client.
Persistent Grants Max Lifetime	Overrides the **Persistent Grant Max Lifetime** field value set globally in the **OAuth Server**# **Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **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 on the **OAuth Server**# **Authorization Server Settings** screen. Once added, the lifetime of persistent grants can be set 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, they can be configured to use the default value.
Persistent Grants Idle Timeout	Overrides the **Persistent Grant Idle Timeout** field value set globally in the **OAuth Server**# **Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **Grants Do Not Timeout Due To Inactivity** ▪ A custom value in days, hours, or minutes. If an idle timeout value is configured, the idle timeout window slides when a persistent grant is updated (see). When an idle timeout value is configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity, or are revoked or removed. When an idle timeout value is configured with a maximum lifetime, persistent grants remain valid until they expire (due to inactivity or lifetime expiration) or are removed from the grant storage.
Client Authentication Certificate Issuer DN	Select a trusted CA from the list. (These are CA certificates imported into PingFederate. You can review them on the **Security**# **Trusted CAs** screen.) Alternatively, you may 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 the **OAuth Server#** **Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **Roll** Note that this selection does not override the **Minimum Interval to Roll Refresh Tokens (Hours)** value set on the **Authorization Server Settings** screen. ▪ **Don't Roll**
OpenID Connect	**Note:** These options are displayed only when the **OpenID Connect** protocol is enabled in **System#** **Protocol Settings#** **Roles & Protocols** screen. **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 either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . **Note:** While all settings on this screen 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, developers can refer to 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	This field controls whether to use global device authorization grant settings defined on the **OAuth Server# Authorization Server Settings** screen. The default selection is **Use Global Settings**. Select **Override** and configure any of the following settings. **User Authorization URL** This field controls whether PingFederate should use a different URL, perhaps for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses (see). 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 (see). Moreover, it 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. **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. This field controls whether PingFederate should skip this confirmation step. Select the **Bypass Activation Code Confirmation** check box if you want PingFederate to skip the confirmation step.

Field	Description
Require Proof Key for Code Exchange (PKCE)	Applicable only when the client is configured to support the authorization code grant type. This field 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 (tools.ietf.org/html/rfc7636). When enabled, this client must include a one-time string value through the use of the code_challenge parameter in its authorization request (see). It must also submit the corresponding code verifier via the code_verifier parameter in its token request when exchanging an authorization code for an access token (see). This check box is not selected by default.
Polling Interval (seconds)	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	The CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication requests; for example, 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 on the **OAuth Server# Request Policies** screen.
Require CIBA Signed Requests	Indicates 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 OpenID Connect specification calls this JWT a *request object*. This check box is not selected by default. Note that if CIBA signed requests are required, the dynamic client registration must include either the JWKS URL or the actual JWKS.

Results

Select client registration policies

About this task

Client registration policies can provide additional control over which registrations and configurations are accepted and stored for each client created via the OAuth 2.0 Dynamic Client Registration protocol. 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 one; otherwise it returns an error message to the developers.

ⓘ **Note:** PingFederate must be able to complete all policies successfully before a client can be created via the OAuth 2.0 Dynamic Client Registration protocol.

Steps

1. Go to the **OAuth Server**# **Client Settings**# **Client Registration Policies** screen.
2. Optional: Select a Client Registration Policy instance from the **Available Policies** list and click **Add**.

   ⓘ **Important:** Although optional, it is recommend to 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** to do so.
3. Optional: Repeat the last step to add other Client Registration Policy instances.

   Add as many Client Registration Policy instances as necessary. Use the up and down arrows to adjust the execution order. Use the **Delete** and **Undelete** workflow to remove an existing instance or cancel the removal request.

Review client settings

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Manage Client Registration Policy instances

About this task

The Client Registration Policy plug-in capability allows you to write custom processing rules to provide additional control over which registrations and configurations are accepted and stored for each client created via the OAuth 2.0 Dynamic Client Registration protocol.

Depending on the complexity of the business or technical requirements of your use cases, you can create one or more Client Registration Policy plug-ins using the PingFederate SDK. After deploying your plug-ins, you can create and configure instances of them. (Note that configuration requirements vary based on your custom solutions.) Finally, 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 located under the `<pf_install>/pingfederate/sdk` directory.

2. Create, modify, or remove one or more instances.
   - To configure a new instance, 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** 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 plug-in configuration, click **Save**.

Results

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 **OAuth Server**# **Client Settings**# **Client Registration Policies** screen.

Configure a Client Registration Policy instance

About this task

Use the **Client Registration Policy** configuration wizard to create a new or modify an existing Client Registration Policy instance.

Steps

1. Go to the **OAuth Server**# **Client Registration Policies** screen.
   - To configure a new instance, click **Create New Instance**.
   - To modify an existing instance, select it by its name under **Instance Name**.
2. On the **Type** screen, enter a name and an ID for a new instance, and then select a plug-in from the list.

   Note that only the name can be changed when modifying an existing policy plug-in instance.

   In addition, if the **Type** list does not contain the desired Client Registration Policy plug-in, 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 located under the `<pf_install>/pingfederate/sdk` directory.
3. On the **Instance Configuration** screen, follow the on-screen instructions to configure the Client Registration Policy instance.

   Note that this screen varies, depending on the selected Client Registration Policy plug-in.
4. On the **Summary** screen, review the plug-in configuration, modify as needed, and click **Done**.
5. On the **Manage Client Registration Policy Instances** screen, click **Save**.

Results

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 **OAuth Server**# **Client Settings**# **Client Registration Policies** screen.

Configure a Response Type Constraints instance

About this task

Configure an instance of the Response Type Constraints policy plug-in 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**

This capability allows administrators to control which flows are allowed for clients created through the OAuth 2.0 Dynamic Client Registration protocol.

For more information about flows and response types, see the *OpenID Connect specification* (openid.net/specs/openid-connect-core-1_0.html#Authentication).

Steps

1. Go to the **OAuth Server# Client Registration Policies** screen.
   - To configure a new instance, click **Create New Instance**.
   - To modify an existing instance, select it by its name under **Instance Name**.
2. On the **Type** screen, enter a name and an ID for a new instance, and then select **Response Type Constraints** from the list.

   Note that only the name can be changed when modifying an existing policy plug-in instance.
3. On the **Instance Configuration** screen, clear the applicable check boxes to remove the unwanted response types.

   (All response types are allowed by default.)
4. On the **Summary** screen, review the plug-in configuration, modify as needed, and click **Done**.
5. On the **Manage Client Registration Policy Instances** screen, click **Save**.

Results

Important: Like other Client Registration Policy plug-ins, an instance of the Response Type Constraints policy plug-in is not enforced (or executed as part of the dynamic client registration process) until it is selected on the **OAuth Server# Client Settings# Client Registration Policies** screen. If it is selected on the **Client Registration Policies** screen, 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.

Manage OAuth clients

About this task

An OAuth client application interacts with an OAuth authorization server (AS) to obtain access tokens needed to call OAuth-protected services at the resource server (RS).

Use the **OAuth Server# Client Management** to manage OAuth clients. This screen displays 20 clients at a time. You can sort the display order by name or ID. 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 so long as its name *or* ID is a partial, case-insensitive match to the search term.

Steps

- To add an client, click **Add Client** and complete the configuration on the **Client** screen.
- To edit a recently modified client, select it by its name and update its configuration on the **Client** screen.
- 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** workflow for the applicable client and then click **Save**.

Results

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 provides the flexibility to manage client records via 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. Instead, they are stored on a database server, a directory server, or some other storage medium through the use of the PingFederate SDK (see *Defining an OAuth client data store* on page 191).

Configure an OAuth client

About this task

The **Client** screen provides controls over the usage and behavior of the applications requesting access to protected resources through the PingFederate OAuth AS.

Steps

1. On the **Manage Client** screen, configure the OAuth client to suit your use cases.

   Refer to the following table for detailed information about each field.

Field	Description
Client ID (Required)	A unique identifier the client provides to the resource server (RS) to identify itself. This identifier is included with every request the client makes.
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 here, then use the same alias in language resource files.

Field	Description
Client Authentication	The authentication method that the client uses. **None** Select this option if your use case does not 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 (see the **Allowed Grant Types** field). ▪ This client signs its ID tokens using an HMAC signing algorithm (see the **ID Token Signing Algorithm** field). ▪ This client is allowed to access the Session Revocation API endpoint (see the **Grant Access to Session Revocation API** field). **Client Secret** Select this option for HTTP Basic authentication. ▪ Click **Generate Secret** to create a strong random alphanumeric string or manually enter a secret. ▪ To modify an existing secret, select the **Change Secret** check box. Then, click **Generate Secret** to create a strong random alphanumeric string or manually enter a secret. **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 from the **Issuer** list. (These are CA certificates imported into PingFederate. You can review them on the **Security# Trusted CAs** screen.) Alternatively, you may 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 (see the property pf.secondary.https.port in the table under *Configuring PingFederate properties* on page 221). **Private Key JWT** Select this option for the private_key_jwt client authentication method, as defined in *Client Authentication* in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication). ▪ Select the **Replay Prevention** check box if PingFederate should mandate a unique signed JWT from the client for each request when the client is configured to authenticate via the private_key_jwt client authentication method, to transmit request parameters using in signed request objects, or to do both. This check box is not selected by default.

Field	Description
Require Signed Request	Indicates 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 OpenID Connect specification calls this JWT a *request object*. ⓘ **Note:** If a client includes in an authorization request a request parameter (other than client_id and response_type) 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 and response_type request parameters, 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. It is also worth noting that if a request parameter is found only outside of the signed request object, such request parameter is dropped and ignored; no error message is returned. ⓘ **Tip:** Per OAuth and OpenID Connect specifications, a client must always include in an authorization request the client_id, response_type, and scope request parameters outside of the signed request object. For more information about request object, please refer to the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#RequestObject).
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 may send its authorization requests using request objects. If PingFederate is either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , 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. Only one of them is required if the client is configured to use the private_key_jwt client authentication method, to transmit request parameters in signed request objects, or to transmit CIBA request parameter in signed request objects (or to do any of them) so that PingFederate can verify the authenticity of the JWTs. In addition, either may also 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 runtime processing, see .) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see .) Finally, if the client is configured to encrypt ID tokens using an asymmetric encryption algorithm (see the **ID Token Key Management Encryption Algorithm** setting), either the JWKS URL or the actual JWKS must be provided.
Redirection URIs	URIs to which the OAuth AS may redirect the resource owner's user agent after authorization is obtained. At least one redirection URI is required by the authorization code and implicit grant types. 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 if a single URI uses wildcards, then 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.)
Bypass Authorization Approval	When selected, resource-owner approval for client access is assumed; 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. Use this setting, for example, when you want to deploy a trusted application and authenticate end users via an IdP adapter or IdP connection.

Field	Description
Restrict Common Scopes	This setting controls whether all existing common scopes and scope groups (and those created in the future) or only the select few should be made available to the client. 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; that is, 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 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. For detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section in *Scopes and scope management* on page 390.
Exclusive Scopes	This setting 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; that is, 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:** 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. For detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section in *Scopes and scope management* on page 390.

Field	Description
Allowed Grant Types	Select at least one grant type that this client is allowed to use. Available grant types are: ▪ **Authorization Code** ▪ **Implicit** ▪ **Refresh Token** ▪ **Client Credentials** ▪ **Device Authorization Grant** ▪ **CIBA** ▪ **Resource Owner Password Credentials** ▪ **Assertion Grants** ▪ **Access Token Validation (Client is a Resource Server)** There is no default selection. (For more information about each grant type, see *Grant types*.)
Restrict Response Types	Select this check box 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** For more information about these response types, see Definitions of Multiple-Valued Response Type Combinations (openid.net/specs/oauth-v2-multiple-response-types-1_0.html#Combinations). The **Restrict Response Type** check box is not selected by default. If selected, you must select at least one allowable response_type parameter value. Additionally, it is worth noting that 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. The following table provides a summary of their relationship.

Response type	Grant types
code	**Authorization Code**
code id_token	**Authorization Code** and **Implicit**
code id_token token	**Authorization Code** and **Implicit**
code token	**Authorization Code** and **Implicit**
id_token	**Implicit**
id_token token	**Implicit**
token	**Implicit**

Field	**Description**
Default Access Token Manager	The default Access Token Management (ATM) instance for this client.
Validate Against All Eligible Access Token Managers	Applicable only to RS clients. If selected, this RS client is not required to specify additional parameters (access_token_manager_id or aud) to disambiguate the ATM instance in its token validation requests. When the RS client does not 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 check box is not selected by default.
Require Proof Key for Code Exchange (PKCE)	Applicable and shown only when the client is configured to support the authorization code grant type. This field 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 (tools.ietf.org/html/rfc7636). When enabled, this client must include a one-time string value through the use of the code_challenge parameter in its authorization request (see). It must also submit the corresponding code verifier via the code_verifier parameter in its token request when exchanging an authorization code for an access token (see). This check box is not selected by default.
Persistent Grants Max Lifetime	Overrides the **Persistent Grant Max Lifetime** field value set globally in the **OAuth Server# Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **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 on the **OAuth Server# Authorization Server Settings** screen. Once added, the lifetime of persistent grants can be set 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, they can be configured to use the default value.

Field	Description
Persistent Grants Idle Timeout	Overrides the **Persistent Grant Idle Timeout** field value set globally in the **OAuth Server# Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **Grants Do Not Timeout Due To Inactivity** ▪ A custom value in days, hours, or minutes. If an idle timeout value is configured, the idle timeout window slides when a persistent grant is updated (see). When an idle timeout value is configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity, or are revoked or removed. When an idle timeout value is configured with a maximum lifetime, persistent grants remain valid until they expire (due to inactivity or lifetime expiration) or are removed from the grant storage.
Refresh Token Rolling Policy	Overrides the **Roll Refresh Token Values** setting configured globally in the **OAuth Server# Authorization Server Settings** screen. Options are: ▪ **Use Global Setting** (the default selection) ▪ **Roll** Note that this selection does not override the **Minimum Interval to Roll Refresh Tokens (Hours)** value set on the **Authorization Server Settings** screen. ▪ **Don't Roll**

Field	Description
OpenID Connect	**Note:** These options are displayed only when the **OpenID Connect** protocol is enabled in **System**# **Protocol Settings**# **Roles & Protocols** screen. **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 either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . **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 (**Direct Encryption with symmetric key**, **AES ... Key Wrap**. and **AES-GCM ... key encryption**) and asymmetric algorithms (**ECDH-ES**, **ECDH-ES ... Key Wrap**, and **RSAES OAEP**). **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 OpenID Connect policy from the list. **Grant Access to Session Revocation API** Select this check box to allow this client application to add sessions to or query the revocation status via the Back-Channel Session Revocation API endpoint at `/pf-ws/rest/sessionMgmt/revokedSris`. Authentication is required. This check box is not selected by default. **Note:** Generally speaking, if clients are allowed to add sessions to the revocation list, 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. (For more information, see .)

Field	Description
Device Authorization Grant	This field controls whether to use global device authorization grant settings defined on the **OAuth Server# Authorization Server Settings** screen. Applicable and shown only if the **Device Authorization Grant** grant type is enabled for the client. The default selection is **Use Global Settings**. Select **Override** and configure any of the following settings. **User Authorization URL** This field controls whether PingFederate should use a different URL, perhaps for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses (see). 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 (see). Moreover, it 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. **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. This field controls whether PingFederate should skip this confirmation step.

Field	Description
CIBA	Applicable and shown only if the **CIBA** grant type is enabled for the client. **Token Delivery Method** The token delivery method supported by the client. 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 call back messages. Required and shown only if ping is the configured token delivery method. **Polling Interval (seconds)** 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** The CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication requests; for example, 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 on the **OAuth Server**# **Request Policies** screen. **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 check box is not selected by default. Note that when user code support is enabled, the associated CIBA request policy must also be user code-enabled. **Require CIBA Signed Requests** Indicates 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 OpenID Connect specification calls this JWT a *request object*.

If you want to enable or disable the client, click the toggle switch.

2. Optional: On the **Extended Properties** screen, add, remove, or update one or more values for any extended properties defined on the **System**# **Extended Properties** screen.

   Applicable and shown only if one or more extended properties are defined on the **System**# **Extended Properties** screen.

   Extended property values can serve as metadata. They can also help drive authentication requirements. To learn more, see .
3. Click **Save**.

Grant mapping

Underneath **OAuth Server**# **Grant Mapping** is where you begin to configure the first stage of the two-stage OAuth attribute mapping process. You can map from authentication sources (IdP adapter instances or IdP connections), authentication policy contracts, or Password Credential Validator instances (for resource owner credentials) to persistent grants. You may define issuance criteria to control whether a fulfillment can be made. Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked or cleaned up.

Manage IdP adapter grant mapping

About this task

Use the **OAuth Server**# **IdP Adapter Mapping** configuration to map values obtained from the authentication source into the persistent grants. 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 on the **OAuth Server**# **Authorization Server Settings** screen, configure a mapping for each as well. You can optionally set up data store queries to supplement values returned from the source. This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

- To create a mapping, select the source of the attributes from the list and click **Add Mapping**.
- To modify an existing mapping, select it by its name under **Mappings**.
- To remove an existing mapping or to 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. In addition, any corresponding entries defined in the **OAuth Server**# **Access Token Mapping** screen will also be removed.

Configure IdP adapter attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, click **Next**.

Fulfill IdP adapter grant mapping

About this task

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grants. 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 on the **OAuth Server**# **Authorization Server Settings** screen, configure a mapping for each as well.

> **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 have two Active Directory domains, the sAMAccountName attribute value of an end user in one domain may collide with that of another end user in the other domain. In this scenario, you can map the Subject DN attribute to the USER_KEY attribute.

Map each attribute from one of these Sources:

- Adapter

  When you make this selection, the associated Value drop-down list contains attributes configured in the IdP adapter instance.
- Context

  Values are returned from the context of the transaction at runtime.

  > **Note:** If `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server**# **Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
  > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.
  >
  >   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 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** as the source and enter a static value.
  >
  >   This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

  > **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.
- Extended Client Metadata

  Values are returned from the client record.
- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used).

- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.
- Text

  The value is what you enter. This can be text only, or you can mix text with references to the attributes returned from the adapter instance, using the syntax:

  `${attribute}`

  You can also enter values from your data store, when applicable, using this syntax:

  `${ds.attribute}`

  where `attribute` is any of the data store attributes you have selected.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for OAuth IdP adapter mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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:** 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   ⓘ **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the IdP adapter mapping

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Configure IdP connection grant mapping

About this task

Use this configuration to map values obtained from the SSO tokens into the persistent grants. 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 on the **OAuth Server# Authorization Server Settings** screen, configure a mapping for each as well. You can optionally set up data store queries to supplement values returned from the source. This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

1. Create a new IdP connection or select an existing IdP connection from the **Service Provider** menu.
2. On the **Connection Type** screen, select the **Browser SSO Profiles** check box and the applicable protocol.
3. On the **Connection Options** screen, select the **Browser SSO** check box and then select the **OAuth Attribute Mapping** check box.

   > **Tip:** You may also select other options on the **Connection Type** and **Connection Options** screens. 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** screen, enter the required information.
5. On the **Browser SSO** screen, click **Configure Browser SSO** and follow its series of tasks to complete the **User-Session Creation** configuration.

6. On the **OAuth Attribute Mapping** screen, 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) on the **User-Session Creation# Target Session Mapping** screen, you may select the **Map to OAuth via Authentication Policy Contract** option, and then select the applicable APC from the list.

Choose an OAuth data store

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, select a data store from the **Active Data Store** list; then click **Next**.

  Follow the **OAuth Attribute Mapping Configuration** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, select **No Data Store**; then click **Next**.

Fulfill OAuth attribute mapping

About this task

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grants. 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 on the **OAuth Server# Authorization Server Settings** screen, configure a mapping for each as well.

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.

Map each attribute from one of the following Sources:

- **AccountLink**

  When selected, the **Value** list is populated with **Local User ID**. Normally, you would map **Local User ID** to an attribute that represents the user identifier; for example, the USER_KEY attribute. In addition, this source appears only if you have elected to use account linking for a target session on the **Identity Mapping** screen.
- **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 is populated with the available context of the transaction. Select the desired context from the list.

  Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

  Note: If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
  - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

    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 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** as the source and enter a static value.

    This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.
- **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 data store. Select the desired attribute from the list.
- **Expression** (when enabled)

  This option provides more complex mapping capabilities; for example, 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 (see **Text**).

  Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
- **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 data store, when applicable, using this syntax:

  `${ds.`*`attribute`*`}`

  where *`attribute`* is any attribute that you have selected from the data store.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.

2. Click **Next**.

Define issuance criteria for OAuth attribute mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See the following table for more information.

Source	Description
AccountLink	Select to evaluate the **Local User ID** value of the user. Visible and applicable only if **Account Linking** is the selected identity mapping method (see).
Assertion	Select to evaluate attributes from the IdP connection.
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.
Extended Client Metadata	Select to evaluate OAuth client metadata.

Source	Description
JDBC, LDAP, or other types of data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)

   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      ⓘ **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the OAuth attribute mapping summary

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Manage authentication policy contract grant mapping

About this task

Use the **OAuth Server**# **Authentication Policy Contract Mapping** configuration to map values obtained from the policy contract into the persistent grants. 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 on the **OAuth Server**# **Authorization Server Settings** screen, configure a mapping for each as well. You can optionally set up data store queries to supplement values returned from the source. This mapping configuration is suitable for the Authorization Code and Implicit grant types.

Steps

- To create a mapping, select the source of the attributes from the list and click **Add Mapping**.
- To modify an existing mapping, select it by its name under **Mappings**.
- To remove an existing mapping or to 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. In addition, any corresponding entries defined in the **OAuth Server**# **Access Token Mapping** screen will also be removed.

Configure policy contract attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, click **Next**.

Fulfill policy contract grant mapping

About this task

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grants. 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 on the **OAuth Server#** **Authorization Server Settings** screen, configure a mapping for each as well.

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.

Map each attribute from one of these Sources:

- Authentication Policy Contract

  When you make this selection, the associated **Value** list consists of the attributes (for example, subject) associated with the APC.
- Context

  Values are returned from the context of the transaction at runtime.

  Note: If `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server#** **Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
  - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

    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 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** as the source and enter a static value.

    This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

  Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

- Extended Client Metadata

  Values are returned from the client record.
- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.
- 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 syntax:

  `${attribute}`

  You can also enter values from your data store, when applicable, using this syntax:

  `${ds.attribute}`

  where `attribute` is any of the data store attributes you have selected.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for policy contract mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See 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:** 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review authentication policy contract mapping

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Manage resource owner credentials grant mapping

About this task

Use the **OAuth Server**# **Resource Owner Credentials Mapping** configuration to map values obtained from the Password Credential Validator instance into the persistent grants. The USER_KEY attribute is the identifier of the persistent grants. If extended attributes are defined on the **OAuth Server**# **Authorization Server Settings** screen, configure a mapping for each as well. You can optionally set up data store queries to supplement values returned from the source. This mapping is intended for the Resource Owner Password Credential grant type.

Steps

- To create a mapping, select the source of the attributes from the list and click **Add Mapping**.
- To modify an existing mapping, select it by its name under **Mappings**.

- To remove an existing mapping or to 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. In addition, any corresponding entries defined in the **OAuth Server**# **Access Token Mapping** screen will also be removed.

Configure resource owner attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, click **Next**.

Fulfill resource owner credentials grant mapping

About this task

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grants. The USER_KEY attribute is the identifier of the persistent grants. If extended attributes are defined on the **OAuth Server**# **Authorization Server Settings** screen, configure a mapping for each as well.

> **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 have two Active Directory domains, the sAMAccountName attribute value of an end user in one domain may collide with that of another end user in the other domain. In this scenario, you can map the Subject DN attribute to the USER_KEY attribute.

Map each attribute from one of these Sources:

- Password Credential Validator

  When you make this selection, the associated Value drop-down list consists of the attributes (for example, username) associated with the credential-validation instance.
- Context

  Values are returned from the context of the transaction at runtime.

> **Note:** If `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server**# **Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
- To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

  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 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** as the source and enter a static value.

  This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

- Extended Client Metadata

  Values are returned from the client record.
- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.
- 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 syntax:

  ```
  ${attribute}
  ```

  You can also enter values from your data store, when applicable, using this syntax:

  ```
  ${ds.attribute}
  ```

  where `attribute` is any of the data store attributes you have selected.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for resource-owner credentials mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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.
Extended Client Metadata	Select to evaluate OAuth client metadata.
JDBC, LDAP, or other types of data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the resource owner credentials mapping

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Token mapping

Underneath **OAuth Server# Token Mapping** is where you begin to configure ID token attribute mapping for OpenID Connect and the second stage of the two-stage OAuth attribute mapping process. For the latter, you can map from persistent grants (and optionally authentication sources, authentication policy contracts, authentication context, or Password Credential Validator instances to access tokens. You may define issuance criteria to control whether a fulfillment can be made.

Access token management

PingFederate supports multiple access token management (ATM) instances. 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.

When defining an ATM instance, you can customize various settings, including the 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. Similarly, you can add a resource server (RS) client to the ACL of multiple ATMs instances, so that only such RS 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 ATM ID (access_token_manager_id) or a resource URI (aud) in their requests to the PingFederate OAuth AS at the authorization endpoint (`/as/authorization.oauth2`), the token endpoint (`/as/token.oauth2`), and the introspection endpoint (`/as/introspect.oauth2`). For RS clients, you may configure on a per-client basis whether an RS client must specify the desired ATM instance in its token validation requests at runtime (see *Configure an OAuth client* on page 416).

At runtime, the PingFederate OAuth AS uses the following rules to determine which ATM instances it should use:

1. Limit 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** screen. For OAuth Assertion Grant requests, it is the set of instances for which a mapping is defined in the IdP connection. Furthermore, the ACL (if configured) can also limits 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 RS clients, you may 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 RS clients do not provide either the access_token_manager_id or aud parameter in their requests and the RS 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.

Manage access token management instances

About this task

Use the **OAuth Server#** **Access Token Management** configuration wizard to specify how the PingFederate OAuth AS manages access tokens.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Define an access token management instance

About this task

Define your access token management instance on the **Type** screen.

Steps

1. Enter an instance name and an instance ID.
2. Choose the plug-in type of the access token management instance from the list.

   Type varies depending on the plug-ins deployed on your server. For information about adding a customized plug-in, please contact a support representative via the *Ping Identity Support Center*.
3. Optional: 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 screens.

Configure an access token management instance

About this task

This configuration varies depending on the type of the access token manager.

Steps

- Refer to subsequent topics for configuration steps.

Configure reference-token management

About this task

Access tokens that use the reference-token data model provide a reference to some set of attributes. The resource server (RS) must de-reference the access tokens for the corresponding identity and security information at the OAuth authorization server (AS) that issued them. (PingFederate is the AS.)

The reference-token data model supports both adaptive clustering and directed clustering. For adaptive clustering, PingFederate 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.

Steps

Modify the default values as needed.

Refer to the following table for detailed information about 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`. 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 in minutes for use with the **Lifetime Extension Policy** setting. 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. This setting limits RPC overhead by suspending the calls until the set threshold is crossed. For example, if the token lifetime is 60 minutes and the threshold is 30%, the lifetime will not be extended until the remaining time is less than 18 minutes. This option could potentially reduce RPC traffic between nodes by orders of magnitude while still supporting a lifetime extension policy. The default value is `30` (percent).
Advanced Fields	
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, it waits for all recipients to respond. The default selection is **Majority of Nodes**.

Field	**Description**
RPC Timeout (Required)	The timeout value (in milliseconds) between cluster nodes during synchronous communication. The recommended value ranges from 100 milliseconds to 1000 (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.

Configure JSON-token management

About this task

JSON Web Token (JWT) bearer access tokens are secure and self-contained tokens. This capability allows the target resource server (RS) to validate the access tokens locally or to send the access tokens to the token issuer (PingFederate as the AS) for validation. The configuration provides for token security using either symmetric keys or asymmetric signing-certificate keys. Multiple entries are allowed for either signing mechanism to facilitate rollover of keys when they expire. This token data model is suitable for both standalone and clustered environments.

Steps

1. Add one or more symmetric keys, signing certificates, or both.

   Click **Add a new row . . .**, enter information, and then click **Update** under **Action**.

   > **Important:** The **Key ID** field values must be unique across all JSON-token management instances, including child instances.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Signing Certificates** and use the **Certificate Management** workflow to complete the task.

   > **Note:** To use an RSA-based algorithm for 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.

2. Change or select entries for required fields and make other changes as needed.

   Refer to the following table for information about each field. For detailed information about the algorithms, please refer to the *JSON Web Algorithms (JWA)* specification (tools.ietf.org/html/rfc7518).

Field	**Description**
Token Lifetime (Required)	The amount of time in minutes that an access token is considered valid. The default value is `120` (minutes).
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 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 .) Required if an asymmetric algorithm is selected in the **JWE Algorithm** list.

Field	Description
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 (**Direct Encryption with symmetric key**, **AES ... Key Wrap**. and **AES-GCM ... key encryption**) and asymmetric algorithms (**ECDH-ES**, **ECDH-ES ... Key Wrap**, and **RSAES OAEP**).
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 JWK format or a certificate. Applicable only if an asymmetric algorithm is selected from the **JWE Algorithm** list. Note that you can only specify an asymmetric encryption key here or the partner's JWKS endpoint in the **Asymmetric Encryption JWKS URL** field.
Asymmetric Encryption JWKS URL	The HTTPS URL of a JSON Web Key Set (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 that 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.
Advanced fields	
Include Key ID Header Parameter	When selected (the default), 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, in the event that the remote server does not contain any cache directives in its response, PingFederate only caches the content for 720 minutes (12 hours). Note that 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 (the default), 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, which 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 removed and left blank, PingFederate does not include the client ID of the requesting client in the self-contained tokens. If clients may use the UserInfo endpoint to retrieve additional claims about the users, refer to *UserInfo endpoint* on page 816 for more information.
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 removed and left blank, PingFederate does not include any scope information in the self-contained token. If clients may use the UserInfo endpoint to retrieve additional claims about the users, refer to *UserInfo endpoint* on page 816 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. (Omitted if left blank.) Additionally, you may extend the contract of the access tokens with an attribute named iss on the **Access Token Attribute Contract** screen. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value you specify on the **Contract Fulfillment** screen overrides the value here.
Audience Claim Value	The value of the Audience claim (aud) in the JWT. (Omitted if left blank.) When no value is specified, PingFederate does not validate the Audience claim (aud) value if any is included in the access token. Additionally, you may extend the contract of the access tokens with an attribute named aud on the **Access Token Attribute Contract** screen. When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the value you specify on the **Contract Fulfillment** screen overrides the value here.

Field	Description
JWT ID Claim Length	Indicates the number of characters of the JWT ID (jti) claim in the JWT. The default value is `0`, which means no claim is included.
Access Grant GUID Claim Name	The name of the JWT claim used to carry the persistent access grant GUID. (Omitted if left blank.) If the claim is present during validation, PingFederate checks the grant database to ensure 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. The path, if specified, must begin with a forward slash; for example, `/oauth/jwks`. The resulting URL is https://<*pf_host*>:<*pf.https.port*>/ext/<*JWKS Endpoint Path*>. Furthermore, the path, if specified, must be unique across all plug-in instances, including any child instances.
JWKS Endpoint Cache Duration	Informs the clients the amount of time 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 (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.
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.

Manage session validation settings

About this task

When an OAuth client presents an access token for validation, PingFederate (as an OAuth authorization server) checks the expiration and the other aspects of the access token. If the validation fails, PingFederate returns an `invalid_grant` error to the client.

When PingFederate authentication sessions are enabled, you can optionally configure the access token validation process to evaluate the authentication sessions of the users (the resource owners) before returning the validation results to the clients. Depending on the feature (or features) selected on the **Session Validation** screen, PingFederate may return to the client an `invalid_grant` error if the associated authentication session has timed out (or expired), is not found, or has been revoked. Moreover, you may also configure PingFederate to extend the authentication sessions upon successful validations.

When any of the session validation features is 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 JWT-based access tokens, the session identifier is part of the access token. With the session identifier, an OAuth client may contact the Session Revocation API endpoint to query the status of an authentication session or to revoke an authentication session.

In essence, the session validation features enable you to conjoin the validity of access tokens and the authentication sessions of the users. Because each feature can be independently enabled or disabled per access token management (ATM) instance, you can fully customize unique API and web SSO experiences for your OAuth clients and thus the users.

ⓘ **Important:** Once any of the session validation features is enabled for an ATM instance, clients will *not* be able to obtain an access token through that ATM instance by presenting a refresh token. Any attempt will result in an `unsupported_grant_type` error. The reason being is that such action, if allowed, defeats the purpose of tying the validity of access tokens and the authentication sessions of the users in the first place. Hence, it follows that the **Session Validation** features are most suitable for clients using the **Implicit** grant type because they do not use refresh tokens. That said, clients using the **Authorization Code** grant type can still take advantage of session validation as needed; they just will not be able to use refresh tokens to obtain access tokens through the ATM instances that have the session validation features enabled.

Steps

1. Go to the **OAuth Server**# **Access Token Management** screen.
2. Select the applicable ATM instance or click **Create New Instance** to create a new ATM instance.

   If you are creating a new ATM instance, provide the required information on the **Type** and **Instance Configuration** screens.

3. On the **Session Validation** screen, select the check box for each relevant feature.

ⓘ **Important:** 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.

If authentication sessions are not enabled, you may still continue selecting features on this screen; however, access token validation may fail until authentication sessions are enabled.

If this is a child instance, select the **Override Session Validation Settings** check box and make the adjustments as needed.

Refer to the following table for information about each feature. Each feature is independent of each other.

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 on the **Sessions** screen. ▪ The authentication session has expired based on the **Max Timeout** field value on the **Sessions** screen. ▪ The authentication session is not found; for example, the user has logged out. You may also optimize the access token lifetime. ▪ If this ATM instance issues internally managed reference tokens, match the value of the **Token Lifetime** (on the previous screen, **Instance Configuration**) to that of the **Idle Timeout**. Similarly, if you choose to specify a **Maximum Token Lifetime** value on the **Instance Configuration** screen, 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 (pi.sri) 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 via 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 on the **Sessions** screen. 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%.

Define the access token attribute contract

About this task

On the **Access Token Attribute Contract** screen, define the attribute contract for the access tokens issued by this access token management (ATM) instance. You must enter at least one attribute. For auditing purposes, an attribute may be chosen as the subject.

Steps

1. Add one or more attributes.

   For JWT bearer access tokens, you may 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 you specify on the **Contract Fulfillment** screen overrides the **Issuer Claim Value** field value (if any) defined on the **Instance Configuration** screen.
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 **Contract Fulfillment** screen overrides the **Audience Claim Value** field value (if any) defined on the **Instance Configuration** screen.
exp	Extends the value of the Expire claim (exp), as defined by the **Token Lifetime** setting on the **Instance Configuration** screen, by the specified value (in seconds).
The **Client ID Claim Name** field value, the **Scope Claim Name** field value, or the **Access Grant GUID Claim Name** field value (if any) defined on the **Instance Configuration** screen of this ATM instance.	When mapping attribute values from authentication sources to the access tokens issued by this ATM instance, the values you specify on the **Contract Fulfillment** screen override the value of the client ID, the scope, or the persistent access grant GUID.

2. Select an attribute from the list under **Subject Attribute Name**.

   When recording OAuth transactions in the audit log, PingFederate populates the subject field with values from this attribute.

Manage resource URIs

About this task

An OAuth client can optionally include the requested resource in a query parameter (aud) when sending its request to the authorization endpoint on the PingFederate OAuth AS.

Specify a list of resource URIs that PingFederate OAuth AS can use to select this access token management instance when the aud query parameter is provided.

Important: The resource URIs must correspond to the resource expected by the resource server (RS).

Steps

- To add a new entry, enter the desired value and click **Add**.
- To modify an existing entry, use the **Edit**, **Update**, and **Cancel** workflow.
- To remove an existing entry, use the **Delete** and **Undelete** workflow.

Results

Define access control

About this task

On the **Access Control** screen, you may restrict which OAuth clients are allowed to use this access token management instance.

Steps

1. Select the **Restrict Allowed Clients** check box.
2. Select a client from the **Allowed Clients** list and click **Add**.

   Repeat this step to select additional clients, as needed.

Results

To removal 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.

Review the access token management configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Manage access token mappings

About this task

In this required configuration, you map attributes to be requested from the OAuth resource server into the access token, the token attribute contract.

When mapping a default context, you define how PingFederate (the OAuth AS) maps values into the attributes based on the persistent-grant USER_KEY and any extended attributes defined on the **OAuth Server# Authorization Server Settings** screen.

When a specific context is selected, you can also map attributes from the selected context, namely the chosen IdP adapter instance, Password Credential Validator instance, authentication policy contract, or IdP connection (with an OAuth attribute mapping configuration or an authentication policy contract mapping configuration) into the access tokens. Additionally, 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 **OAuth Server# Access Token Mapping** configuration wizard becomes available only after at least one Access Token Management (ATM) instance has been configured on the **OAuth Server# Access Token Management** screen.

Steps

- To 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**.
- To modify an existing mapping, select it by its name under **Mappings**.
- To 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.

Configure access token attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, click **Next**.

Configure access token fulfillment

About this task

On the **Contract Fulfillment** screen, you map values into the token attribute contract. These are the attributes that will be included or referenced in the access token.

Map each attribute to fulfill the Token Attribute Contract from one of these Sources:

- Client Credentials, IdP Adapter, IdP Connection, or Password Credential Validator

  If you have selected an IdP adapter instance, an IdP connection, or a Password Credential Validator instance under **Context** on the **Access Token Attribute Mapping** screen, you have the option to map attributes from that specific authentication system. Select the corresponding context under **Source** and the desired attribute under **Value**.
- Persistent Grant

  When you make this selection, the associated **Value** list is populated by 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 more appropriate to evaluate and return values.

  Select **Expression** under **Source** and then click **Edit** to enter an expression.

  Additionally, you can use an expression to retrieve from the **HTTP Request** Java object the authentication method that a client uses (or the private key JWT with which a client authenticates if the client uses the private_key_jwt authentication method). For sample expressions, see *Expressions for OAuth and OpenID Connect uses cases* on page 863.

  (If the **Expression** selection is not available, you may 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 (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.
- 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 syntax:

  `${USER_KEY}`

  You can also enter values from your data store, when applicable, using this syntax:

  `${ds.attribute}`

  where `attribute` is any of the data store attributes you have selected.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for access token mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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.
Extended Client Metadata	Select to evaluate OAuth client metadata.
JDBC, LDAP, or other types of data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.
Mapped from Context (Adapter, Authentication Policy Contract, IdP Connection, or Password Credential Validator)	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).
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**.

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

   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 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.
4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.
5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the access token mapping

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Configure an OAuth assertion grant IdP connection

About this task

An OAuth assertion grant connection exchanges a SAML assertion or a JWT for an OAuth access token with the PingFederate OAuth AS. You can configure an OAuth assertion grant connection with an IdP partner either in conjunction with browser-based SSO, WS-Trust, or independently.

For more information about these standards, see *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants* (tools.ietf.org/html/rfc7522) and *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants* (tools.ietf.org/html/rfc7523).

Steps

1. On the **Connection Type** screen, select the **OAuth Assertion Grant** check box.

   Tip: You may also select other options (for example, the **Browser SSO Profiles** check box). If you do, you will be prompted to complete the required configuration.

   For simplicity, this topic only focuses on the **OAuth Assertion Grant** configuration.
2. On the **General Info** screen, enter the required information.
3. On the **OAuth Assertion Grant Attribute Mapping** screen, click **Configure OAuth Assertion Grant Attribute Mapping**.

Define 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** screen.

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:
  a. Enter the attribute name in the text box.

     Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.
  b. Select the check box under **Mask Values in Log**.
  c. Click **Add**.

- To modify an attribute name or masking selection, follow these steps:
  a. Click **Edit** under **Action** for the attribute.
  b. 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.

Configure access token manager mappings

About this task

Use the **OAuth Assertion Grant Attribute Mapping**# **Access Token Manager Mapping** screen to associate one or more access token manager instances with this connection to define how access tokens are created.

Steps

- To create a new Access Token Manager mapping configuration, click **Create New Access Token Manager Mapping**.
- To edit an existing Access Token Manager mapping configuration, select the mapping configuration by its name.
- To delete an Access Token Manager mapping, select **Delete** under **Action** for the applicable mapping configuration.

Select an access token manager instance

About this task

On the **OAuth Assertion Grant Attribute Mapping**# **OAuth Assertion Grant Attribute Mapping Configuration**# **Access Token Manager** screen, select an Access Token Manager instance to associate with this connection. This configuration maps attribute values from the IdP connection into the access token to define the resulting content of the access token.

Steps

- Select an Access Token Manager instance from the list.

> **Tip:** If the Access Token Manager instance you need is not available, click **Manage Access Token Management Instances** to define one or more instances you need for this connection.

Configure a data store for OAuth assertion grant attribute mapping

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, select a data store from the **Active Data Store** list; then click **Next**.

  Follow the configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, select **No Data Store**; then click **Next**.

Configure OAuth assertion grant contract fulfillment

About this task

On the **OAuth Assertion Grant Attribute Mapping**# **OAuth Assertion Grant Attribute Mapping Configuration**# **Contract Fulfillment** screen, map values from the SAML assertions or JWTs to the attributes defined for the attribute contract. These are the values that the Access Token Manager instance requires to create an OAuth access token.

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

Map attributes from one of the following Sources:

- **Assertion**

  When selected, the **Value** list is populated 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 is populated with the available context of the transaction.

> **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values (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 data store. Select the desired attribute from the list.
- **Expression** (when enabled)

  This option provides more complex mapping capabilities; for example, 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 (see **Text**).

  Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
- **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 data store, when applicable, using this syntax:

  `${ds.attribute}`

  where *`attribute`* is any attribute that you have selected from the data store.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for OAuth assertion grant

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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:** 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

   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 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.
4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.
5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review OAuth assertion grant attribute mapping configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review OAuth assertion grant configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure OpenID Connect policies

About this task

This configuration allows you to define OpenID Connect policies for client access to attributes mapped according to OpenID specifications. It also provides an option to include a session identifier in the ID Tokens, which could be useful for the relying parties, such as PingAccess® for client session management.

Steps

- To configure a new OpenID Connect policy, click **Add Policy**.
- To modify an existing OpenID Connect policy, select it by its name under **Policy ID**.
- To review the usage of an existing OpenID Connect policy, click **Check Usage** under **Action**.
- To remove an existing OpenID Connect policy or to cancel the removal request, click **Delete** or **Undelete** under **Action**.
- To elect an existing OpenID Connect policy to be the default OpenID Connect policy, click **Set as Default** under **Action**.

Configure policy and ID token settings

About this task

On the **Manage Policy** screen, enter the required information and configure optional settings for ID tokens issued under this policy.

Steps

1. Enter the policy identifier in the **Policy ID** field.
2. Enter the policy name in the **Name** field.
3. Select an access token management instance from the **Access Token Manager** list.

4. Optional: Define the expiry information (in minutes) for ID tokens issued based on this policy in the **ID Token Lifetime** field.

   The default value is `5` (minutes).
5. Optional: Select the **Include Session Identifier in ID Token** check box to add a session identifier (pi.sri) in the ID tokens.
6. Optional: Select the **Include User Info in ID Token** check box to include additional attributes in the ID tokens.

   > ⓘ **Tip:** Alternatively, OAuth clients can obtain additional attributes from the UserInfo endpoint at `/idp/userinfo.openid` (see *UserInfo endpoint* on page 816).
7. 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, refer to *Financial Services – Financial API - Part 2: Read and Write API Security Profile* from OpenID Foundation (openid.net/specs/openid-financial-api-part-2.html).

Configure the policy attribute contract

About this task

On the **Attribute Contract** screen, define the list of attributes that PingFederate can return to the OAuth clients. Every new OpenID Connect policy contract begins with a list of standard attributes. These are attributes (or claims) defined in the OpenID Connect specification. You can optionally remove standard attributes, edit them to turn them into non-standard attributes, and add new non-standard attributes.

> ⓘ **Note:** In OpenID Connect, scopes affect the list of attributes that PingFederate can return to the OAuth clients. In other words, the attributes that PingFederate returns to OAuth clients vary, depending on the scopes approved by the resource owner in the first place.

By default, all attributes defined on this screen are deliverable through the UserInfo endpoint. In the scenario where an implicit client makes a token request by providing `id_token` as the sole response_type parameter value, the client will only receive an ID token without an access token. Because 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 on the **Manage Policy** screen for this policy, you may 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:
  a. Enter the name of the attribute under **Extend the Contract**.
  b. Optional: Select the **Override Default Delivery** check box to choose how the attribute is delivered.
     - Select the check box under **ID Token** if this attribute can be included in ID tokens.
     - Select the check box under **UserInfo** if this attribute can be included in UserInfo responses.
  c. Click **Add**.
- To modify an existing entry, use the **Edit**, **Update**, and **Cancel** workflow. Choose how the attribute is delivered, as needed.
- To remove an existing entry, click **Delete**.

Configure attribute scopes

About this task

In OpenID Connect, scopes affect the list of attributes that PingFederate can return to the OAuth clients. On the **Attribute Scopes** screen, you can optionally add associations between scopes and attributes beyond what is defined in the specification.

Steps

1. Optional: On the **Attribute Scopes** screen, add any number of scope-to-attributes associations.
   a. Select a scope from the list.

      Both common and exclusive scopes are available for selection.
   b. Select the relevant check boxes under **Attributes**.

      Note: If you have selected a standard scope in the previous step, its associated standard attributes, as defined in the OpenID Connect specification, are automatically selected and cannot be modified. You can however select additional attributes to be associated with the selected scope.

      Additionally, 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. For your convenience, the administrative console detects this condition and 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 if they shall remain inaccessible for the time being.
   c. Click **Add**.
   d. Optional: Repeat these steps to define additional scope-to-attributes associations.

      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.
2. Click **Next**.

Configure policy attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.
- To skip this optional configuration, click **Next**.

Configure ID token fulfillment

About this task

On the **Contract Fulfillment** screen map attributes from the access token or other sources to fulfill the attribute contract.

Map the subject attribute and all extended attributes from one of these Sources:

- 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 more appropriate to evaluate and return values.
  >
  > Select **Expression** under **Source** and then click **Edit** to enter an expression.
  >
  > (If the **Expression** selection is not available, you may 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 (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries are also available for expressions.
- 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 syntax `${attribute}`.

  You can also enter values from your data store, when applicable, using this syntax:

  `${ds.attribute}`

  where `attribute` is any of the data store attributes you have selected.
- Access Token

  The value is provided from the access token.

Steps

1. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Define issuance criteria for policy mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

> ⓘ **Note:** Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See the following table for more information.

Source	Description
Access Token	Select to evaluate attributes from the access token.
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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 a variety of ways; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review your OpenID Connect policy

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Results

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 **Policy Management** screen. You can change the default as needed when you create additional policies.

Client Initiated Backchannel Authentication (CIBA)

Client Initiated Backchannel Authentication (*openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html*) is an extension to OpenID Connect that is gaining interest by organizations that want to improve the end-user experience during authentication and authorization in a federated environment. This extension defines a new OAuth grant type where user consent can be requested through an out-of-band flow. For example, CIBA improves the user experience when making an online purchase from a merchant as 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.

A CIBA configuration consists of two components.

CIBA authenticator

A CIBA authenticator is responsible for authenticating users through an out-of-band method.

You may use the PingFederate SDK to implement a custom solution. For more information, refer to 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.

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.

Configuration steps

1. Create an instance of a CIBA authenticator.
   a. Open the **OAuth Server**# **Authenticators** screen.
   b. Click **Create New Instance**.
   c. On the **Type** screen, provide the required information and select a CIBA authenticator from the **Type** list.

   Note: Selections vary depending on the CIBA authenticators that have been installed in your PingFederate environment.

   d. Click **Next** to access the **Instance Configuration** screen.

   From this point forward, follow the on-screen instructions to complete the configuration. For more information, see *Configure a CIBA authenticator instance* on page 472.
2. Create a CIBA request policy.
   a. Open the **OAuth Server**# **Request Policies** screen.
   b. Click **Add Policy**.
   c. On the **Manage Policy** screen, provide the required information, including selecting the CIBA authenticator instance created in the previous step from the **Authenticator** list.

   From this point forward, follow the on-screen instructions to complete the configuration. For more information, see *Define a request policy* on page 473.
3. Create an OAuth client.
   a. Open the **OAuth Server** menu.
   b. Click **Create New** under **Clients**.
   c. On the **Client** screen, provide the required information.

   To enable CIBA for the client, you must select **CIBA** in the **Allowed Grant Types** setting. Once selected, you can configure a few more client CIBA-related settings.

   For more information, see *Configure an OAuth client* on page 416.

Manage CIBA authenticators

About this task

A CIBA authenticator is responsible for authenticating users through an out-of-band method.

You may use the PingFederate SDK to implement a custom solution. For more information, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.

Once deployed, you can create one or more instance configurations of the authenticator.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Configure a CIBA authenticator instance

About this task

Steps

1. Click **OAuth Server# Authenticators** to open the **CIBA Authenticators** screen.
2. On the **CIBA Authenticators** screen, click **Create New Instance** to start the **Create CIBA Authenticator Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this authenticator instance.
   a. Enter a name and an ID in the **Instance Name** and **Instance ID** fields.
   b. Select a CIBA authenticator from the **Type** list.

      Selections vary depending on the deployed CIBA authenticators.

      You may use the PingFederate SDK to implement a custom solution. For more information, refer to 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 located under the `<pf_install>/pingfederate/sdk` directory.
4. On the **Instance Configuration** screen, follow the on-screen instructions to configure the authenticator instance.

   Configuration requirements vary depending on the authenticator implementation.
5. On the **Actions** screen, follow the on-screen instructions to test the validity of the authenticator instance configuration or to perform secondary configuration tasks.

   Availability of this screen and actions vary depending on the authenticator implementation.

6. On the **Extended Contracts** screen, 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 screen and supported attribute names vary depending on the authenticator implementation.
7. On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Create CIBA Authenticator Instance** workflow.
8. On the **CIBA Authenticators** screen, click **Save** to retain the configuration of the authenticator instance.

   If you want to exit without saving the configuration, click **Cancel**.

Manage CIBA request policies

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

- To configure a new CIBA request policy, click **Add Policy**.
- 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**.

Define a request policy

About this task

Steps

1. On the **Manage Policy** screen, 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.

Field	**Description**
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 has the option to support user code in its configuration. However, a client supporting user code must not be associated with a request policy that has not been configured with a PCV instance. For more information on CIBA client configuration, see *Configure an OAuth client* on page 416.
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 acknowledgements. The default value is `120`. Note that clients have the option to 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 JWTs 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.

2. Click **Next**.

Configure identity hint contract

About this task

The identity hint contract is the set of attributes received in the CIBA request that identifies the user. 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** screen.

Steps

1. Optional: Enter an attribute name under **Extend the Contract** and then click **Add**.
2. Repeat the previous step to define additional attributes.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing entry. Click **Delete** to remove an entry.
3. Click **Next**.

Example

Suppose the following 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.

Configure identity hint contract fulfillment

About this task

On the **Identity Hint Contract Fulfillment** screen, you may process the identity hint to further augment the identity data prior to contract fulfillment.

Steps

1. Click **Manage Fulfillment** to begin the mapping configuration.
2. When the administrative console returns you to the **Identity Hint Contract Fulfillment** screen, click **Next**.

Configure attribute sources and user lookup

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration to fulfill the identity hint's attribute contract is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.

  When the administrative console returns you to the **Attribute Sources & User Lookup** screen, click **Next**.
- To skip this optional configuration, click **Next**.

Fulfill identity hint contract

About this task

On the **Identity Hint Contract Fulfillment Mapping** screen, fulfill the identity hint contract with values from the original identity hint, data stores, dynamic text values, or attribute mapping expressions (if enabled).

Steps

1. Select a source from the list.

   For more information about the **Source** list, refer to the following table.

Source	Description
Context	Select **Context** to return specific information from the request.
JDBC, LDAP, or other types of data store (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a data store. When you make this selection, the list under **Value** is populated with attributes from your database, directory, or other data store. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** screen (see *Configure attribute sources and user lookup* on page 475).
Request	Select **Request** to use the attribute value PingFederate found in the CIBA request without customization.
Expression (if enabled)	Select **Expression** to support complex mapping requirements; for example, 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* on page 859.
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 may 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 the **Attribute Sources & User Lookup# Data Store** screen and `attribute` is an attribute from data store.

2. Specify a value associated with the selected source.

   Inapplicable if you have selected **Request** from the **Source** list.
3. Repeat these steps until all attributes are configured.
4. Click **Next**.

Define issuance criteria for identity hint contract

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

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 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.

4. Enter the desired (compared-to) value under **Value**.

> **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.

7. Optional: Repeat to add multiple criteria using the user interface.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)

a. Click **Show Advanced Criteria**.

b. Enter the required expressions in the **Expression** field.

c. Optional: Enter an error code or an error message in the **Error Result** field.

> **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

d. Click **Add**.

e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.

f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

9. Click **Next**.

Review identity hint contract fulfillment

About this task

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.
- To discard your changes, click **Cancel**.

Configure attribute sources and user lookup for request policy contract

About this task

You can optionally set up data store queries to supplement values returned from the source. This configuration to fulfill the request policy's attribute contract is optional.

Steps

- To set up data store queries, click **Add Attribute Source**.

  Follow the configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 872.

  When the administrative console returns you to the **Attribute Sources & User Lookup** screen, click **Next**.
- To skip this optional configuration, click **Next**.

Configure request policy contract fulfillment

About this task

On the **Contract Fulfillment** screen, fulfill the request policy contract with values from the original identity hint, data stores, dynamic text values, or attribute mapping expressions (if enabled).

This contract is used to map into the OAuth grant (the USER_KEY attribute), the CIBA authenticator (attributes vary depending on the authenticator), and the user code PCV (the USER_CODE_USER_NAME attribute). The USER_CODE_USER_NAME atttribute is shown only if a PCV instance is selected on the **Manage Policy** screen.

Steps

1. Select a source from the list.

   For more information about the **Source** list, refer to the following table.

Source	Description
Context	Select **Context** to return specific information from the request.
JDBC, LDAP, or other types of data store (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a data store. When you make this selection, the list under **Value** is populated with attributes from your database, directory, or other data store. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** screen (see *Configure attribute sources and user lookup for request policy contract* on page 479).
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; for example, 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* on page 859.
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 may 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 the **Attribute Sources & User Lookup# Data Store** screen and `attribute` is an attribute from data store.

2. Specify a value associated with the selected source.
3. Repeat these steps until all attributes are configured.
4. Click **Next**.

Define issuance criteria for CIBA request policy

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.
9. Click **Next**.

Review your CIBA request policy

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Results

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 Policy Management** screen. You can change the default as needed when you create additional policies.

OAuth attribute mapping using a data store

About this task

This optional configuration is the same for all OAuth grant mapping and token mapping configurations (see *Grant mapping* on page 427 and *Token mapping* on page 445, respectively).

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

About this task

Asynchronous Front-Channel Logout provides OAuth clients the capability to initiate single logout requests to sign off associated SLO-enabled 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 OpenID Connect metadata at the `/.well-known/openid-configuration` endpoint. Look for `ping_end_session_endpoint` in the metadata.

On a per-client basis, PingFederate can be configured to send (via the browser) logout requests to PingAccess® and additional requests to other relying parties.

When the PingAccess option is selected, PingFederate sends logout requests (via the browser) to the OpenID Connect 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.

In addition, when signing off an SLO-enabled SAML 2.0 or WS-Federation session, as the SP-initiated logout request reaches the PingFederate IdP server, the same logout process applies as well. Depending on the enterprise architecture, this could further improve single sign-on and logout use cases.

Back-Channel Session Revocation

About this task

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.

Identity provider SSO configuration

In an IdP role, you use the PingFederate administrative console to configure local application-integration information and to manage connections to your SP-partner sites. Prior to configuring connections to SPs, you must establish your site as an IdP on the **System# Protocol Settings# Roles & Protocols** screen.

Note that only one connection is needed 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 **System# Protocol Settings# Federation Info** screen, you may identify your organization differently through the use of virtual server IDs on a per-connection basis (see *Multiple virtual server IDs* on page 133).

Additionally, you may deploy an SP connection to bridge a service provider to one or more identity providers through one or more authentication policy contracts (see *Federation hub use cases* on page 127).

Note: This topic applies to configuration settings needed for browser-based SSO. While there is some cross-over information also applicable to WS-Trust STS, if you are using PingFederate *exclusively* as an STS, start with *WS-Trust STS configuration* on page 673.

IdP application integration settings

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 Software Development Kit (see *SSO integration kits and adapters* on page 112).

Under **Integration** on the **Identity Provider** menu, you configure instances of the IdP adapters (for example, the HTML Form Adapter) that PingFederate needs to interact with applications or access-management systems used to authenticate users at your site. For authentication sources and selectors that are capable of delegating end-user interactions to external web applications, you may create authentication applications to represent them. You can also set a Default URL to which users may be directed during SLO, and you can 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 STS, the selections under **Integration** also include menu items for configuring plug-in **Token Processors** and optionally **STS Request Parameters** (see *IdP configuration for STS*).

Manage IdP adapters

About this task

An IdP adapter is used to look up session information and provide user identification to PingFederate. You must configure at least one instance of an IdP adapter in order to set up connections to SP partners.

PingFederate comes bundled with the PingID® integration kit and the following adapters:

- HTML Form Adapter
- HTTP Basic Adapter
- Kerberos Adapter
- OpenToken Adapter
- Composite Adapter
- PingID Adapter

You can also deploy additional integration kits from the Ping Identity Downloads website.

You manage IdP adapter instances on the **Identity Provider# Adapters** screen.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Results

ⓘ **Note:** Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.

If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

Create an IdP adapter instance

About this task

The first step in creating an adapter instance is choosing an adapter type.

Steps

- On the **Type** screen, configure the basics of this adapter instance.
  a. Enter the required information and select the adapter type from the list.
  b. Optional: 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 screens.

Configure an IdP adapter instance

About this task

Depending on the selected adapter, the **IdP Adapter** screen presents you with different configuration parameters.

Steps

- Follow the on-screen instructions to configure the adapter instance.

  ⓘ **Note:** If this is a child instance, select the override check box to modify the configuration.

  If you are configuring one of the bundled adapters, refer to *Bundled adapters* on page 709 or *PingID® for PingFederate SSO* for configuration information.

  If you are configuring an adapter from an integration kit (including any SaaS connector), locate the user guide from our *PingFederate documentation* website and configure the adapter instance accordingly.
- Click **Show Advanced Fields** to enter an authentication context URI.

  Applicable only if the adapter supports the notion of authentication context.

  ⓘ **Tip:** Standard URIs are defined in the SAML specifications (see the OASIS documents *oasis-sstc-saml-core-1.1.pdf* and *saml-authn-context-2.0-os.pdf*).

Invoke IdP adapter actions

About this task

Adapters may be written to provide configuration assistance or validation actions. Actions may also include generation of parameters that might need to be set manually in a configuration file.

Steps

- Follow the on-screen instructions to complete the actions required.

Extend an IdP adapter contract

About this task

You can extend the IdP adapter contract with additional attributes in the **Extended Contract** screen. For the Composite Adapter, attributes from the IdP adapter instances that comprise the composite configuration *must* be added on this screen.

If the adapter does not return values for the extended attributes (or if you prefer to fulfill them differently using data store queries, dynamic text values, or results from OGNL expressions), you can define their fulfillment on the **Adapter Contract Mapping** screen later in the connections.

Note: If this is a child instance, select the override check box to modify the configuration.

Steps

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Set pseudonym and masking options

About this task

Steps

- On the **Adapter Attributes** screen, configure the pseudonym and masking options.

  Note: The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

  a. 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 SP partners use pseudonyms for account linking.

     Note: A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.
  b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
  c. 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

Define the IdP adapter contract

About this task

An IdP adapter contract is a contract that may be used to fulfill the attribute contract passed to your SP partners. By default, PingFederate fulfills the IdP adapter contract with attribute values from the adapter. Optionally, you can configure PingFederate to fulfill the IdP adapter contract with attribute values from local data stores, dynamic text values, results from OGNL expressions, or a combination of them. In addition, you have the option to verify requests using the Token Authorization framework.

To customize the IdP adapter contract,

Steps

1. Select the applicable IdP adapter instance.
2. Go to the **Adapter Contract Mapping** screen.
3. **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** screen, in which case the **Override Adapter Contract** check box is automatically selected for you.

   Click **Configure Adapter Contract**.

Define attribute sources and user lookup

About this task

Attribute sources are specific data store or directory locations containing information that may 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.

The PingFederate IdP server supports separate data stores to look up attributes for a single mapping. For example, you can query multiple data stores for values and map those values in one mapping, or query a data store for a value and use that value as input for subsequent queries of other data stores.

Queries are executed in the order as shown on the **Attribute Sources & User Lookup** screen. Use the up and down arrows as needed to adjust the order.

If a required attribute (such as the user identifier username or subject for the HTML Form Adapter or the OpenToken IdP Adapter, respectively) cannot be fulfilled, the request fails.

Steps

- If your use case requires only dynamic texts or results from OGNL expressions without any attributes from local data stores, skip to the next screen.
- To add an attribute source, click **Add Attribute Source**.

  Repeat this step as needed to add another attribute source.
- To modify an existing instance, select it by its name under **Description**.
- To remove an existing instance or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

Configure IdP adapter contract fulfillment

About this task

On the **Adapter Contract Fulfillment** screen, map values into the IdP adapter contract.

Steps

1. Select a source from the list.

   For more information about the **Source** list, refer to 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.

Source	Description
JDBC, LDAP, or other types of data store (if configured)	Select an attribute source when PingFederate should retrieve attribute value from a data store. When you make this selection, the list under **Value** is populated with attributes from your database, directory, or other data store. Applicable only if you have added at least one attribute source on the **Attribute Sources & User Lookup** screen (see *Define attribute sources and user lookup* on page 487).
Expression (if enabled)	Select **Expression** to support complex mapping requirements; for example, 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* on page 859.
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 may 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 the **Attribute Sources & User Lookup# Data Store** screen and `attribute` is an attribute from data store.

2. Specify a value associated with the selected source.

   Inapplicable if you have selected **Adapter** from the **Source** list.
3. Repeat these steps until all attributes are configured.

Define issuance criteria for IdP adapter contract

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

> **Note:** Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   > ⓘ **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      > ⓘ **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review an IdP adapter contract

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review and save an IdP adapter configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

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 one of the browser-based 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 steps through 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 action ID is specified via the Content-Type HTTP request header in this format: `application/vnd.pingidentity.`*`<actionId>`*`+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 may also be sent if PingFederate encounters an 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.

Recap of 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 PF 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 PF session cookies, only browser-based web applications can currently make use of the API. Server-side web applications are not supported.

To ensure that the PF session cookies are managed correctly, ensure the Javascript-written authentication application sets 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. (Note that the value of this header does not matter.)

Default authentication application

In addition to specifying an authentication application for each policy, you can also configure a default application. The default application is used if an API-capable adapter is encountered outside of a policy tree. For example, the default application is used if the user navigates to the Change Password endpoint (`/ext/pwdchange/Identify`) or the Account Recovery endpoint (`/ext/pwdreset/Identify`). The default application is also used if authentication policies are disabled, or if the flow falls through to a default authentication source.

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`. For more information, see *Explore authentication API* on page 493.

Manage authentication applications

About this task

Authentication applications present user interfaces to collect credentials when authentication is completed via the PingFederate authentication API. The default authentication application is used for authentication sources that are invoked directly (not as part of an authentication policy) and that support the authentication API functionality.

Steps

- To toggle the availability of authentication API support, select or clear the **Enable Authentication API** check box.

  This check box is not selected by default.
- To toggle the availability of the Authentication API Explorer, select or clear the **Enable API Explorer** check box.

  Applicable and shown if the **Enable Authentication API** check box is selected.

  When shown, this check box is selected by default.
- To set a default authentication application, choose an authentication application from the **Default Authentication Application** list.
- To not set a default authentication application, choose **Select** from the **Default Authentication Application** list.
- To add an authentication application, click **Add Authentication Application**.
- To remove an authentication application or to cancel the removal request, click **Delete** or **Undelete** under **Action** for the applicable authentication application.

Configure an authentication application

About this task

On the **Authentication Application** screen, create a new or manage an existing authentication application.

Steps

1. Go to **Authentication# Integration# Authentication API Applications**.
   - To create a new application, click **Add Authentication Application**.
   - To modify an existing application, click the **Application Name**.
2. In the **Authentication Application** window, 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.
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, please refer to *W3C's recommendation on Cross-Origin Resource Sharing* (www.w3.org/TR/cors).

Sample entry	Expected behavior
`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: While 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. Multiple entries are allowed.

3. Click **Save** to keep your configuration or click **Cancel** to discard any changes made.

Explore authentication API

About this task

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.

Steps

Enable Authentication API

1. On the **Identity Provider# Authentication Applications** screen, select the **Enable Authentication API** check box.

Enable API Explorer

2. On the **Identity Provider# Authentication Applications** screen, select the **Enable API Explorer** check box.

Configure an authentication application for the Authentication API Explorer

3. On the **Identity Provider# Authentication Applications** screen, click **Add Authentication Application**.
4. On the **Authentication Application** screen, configure each field as described in the following table.

Field	Description
Name	A name of the authentication application; for example: `Authentication API Explorer`
Description	An optional description of the authentication application; for example: `Explore the authentication API!`
URL	A combination of the PingFederate base URL and the application path of /pf-ws/authn/explorer For example, if the base URL is https://localhost:9031, enter: `https://localhost:9031/pf-ws/authn/explorer`
Additional Allowed Origins	Any additional allowed origins (see *Configure an authentication application* on page 492). If you are using a PingFederate base URL of https://localhost:9031 for testing purpose, you can skip this field.

5. On the **Authentication Application** screen, click **Save**.
6. On the **Manage Authentication Applications** screen, click **Save** as well.

Explore the available states, actions, and models for any API-capable adapter or selector in your PingFederate environment

7. Browse to the URL of the Authentication API Explorer; for example:

   https://localhost:9031/pf-ws/authn/explorer
8. Select an adapter or authentication selector from the **Authentication Source/Selector** list.

   Based on your selection, 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

   This information is vital for the developers of your web applications to create the desired authentication experience.

Explore the authentication API through a request without an authentication policy

9. Configure a use case to use an instance of the HTML Form Adapter for authentication.

   For example, you may create an SP connection that uses an instance of the HTML Form Adapter for authentication.
10. On the **Identity Provider# Authentication Applications** screen, choose the authentication application that represents the Authentication API Explorer from the **Default Authentication Application** list.
11. Initiate a request supported by the use case you configured in step 9.

    When PingFederate receives your request, it gathers 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 screen (from the HTML Form Adapter), PingFederate redirects the browser to the Authentication API Explorer with a flowid query parameter; for example: 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, `Tt9n7` in our example.

12. In the Authentication API Explorer, click **Get** next to the pre-populated flow ID value.

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 is required).

Explore the authentication API through a request with an authentication policy

13. On the **Identity Provider# Authentication Applications** screen, choose **Select** from the **Default Authentication Application** list.

As a result, there is no default authentication application now.

14. Define an authentication policy to use the Authentication API Explorer.

a. On the **Identity Provider# Policies** screen, click **Add Policy**.
b. Enter a policy name and optionally a description.
c. Select the authentication application that represents the Authentication API Explorer from the **Authentication Application** list.
d. Select the HTML Form Adapter instance that has been mapped to your use case in step 9.

   For both the **Fail** and **Success** policy paths, select **Done**.
e. Click **Done**.
f. Select the **IdP Authentication Policies** check box.
g. Click **Save**.

15. Initiate a request supported by the use case you configured in step 9.

16. Use the Authentication API Explorer to learn more about the authentication API.

Configure a default URL and error message

About this task

As an IdP, you can specify to prompt end users to confirm their single logout requests and a default URL indicating a successful SLO to the end-user (if no other page is designated). On the **IdP Default URL** screen, 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 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.

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* on page 275. If localization is not needed, you may also specify a message directly in this field to change the default.

Your application or your partner's application may supply the URL at runtime (see *IdP endpoints* on page 757). However, if none is provided, PingFederate will use the default value you enter on this screen.

ⓘ **Tip:** If you leave the default URL blank, PingFederate provides built-in landing page for the user. This web page is among the templates you can modify with your own branding or other information (see *Customizable user-facing screens* on page 257).

View IdP application endpoints

About this task

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate.

Steps

- Click **Application Endpoints** on the **Identity Provider** menu 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* on page 757 and *System-services endpoints* on page 773.

View IdP protocol endpoints

Click **Protocol Endpoints** on the **Identity Provider** menu to see a list of applicable SAML, WS-Federation, and WS-Trust STS endpoints. The pop-up window displays only those endpoints related to the federation protocols enabled on the **System# Protocol Settings# Federation Info** screen. These endpoints are built into PingFederate and cannot be changed.

Your federation partners or 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 favorite icon for all protocol endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 291.

The table below 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 **Important** 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 **Important** note at the end of this table.)

Service	**URL and Description**
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 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 **Important** 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 SLO messages.
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 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 (see). **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—see . (See also "**Important**" footnote in 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` (see).

Virtual server ID support

For SAML connections using multiple virtual server IDs (see *Multiple virtual server IDs* on page 133), each virtual server ID has its own set of protocol endpoints. You may export a connection metadata for your partner on the **System**# **Metadata Export** screen (see *Exporting connection-specific SAML metadata* on page 155).

For WS-Federation (and SAML) connections using multiple virtual server IDs, you may provide your partner the federation metadata endpoint (`/pf/federation_metadata.ping`) with the PartnerSpId *and* vsid parameters; for 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.)

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 may provide your partner the STS metadata endpoint (`/pf/sts_mex.ping`) with the PartnerSpId *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* on page 773.)

Note that 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 may pass the token-to-token endpoint to your partners as-is.

Manage SP connections

As an IdP, you manage connection settings to support the exchange of federation-protocol messages (SAML, WS-Federation, or WS-Trust) with an SP or STS client application at your site.

These settings include:

- User attributes that you expect to send in an SSO token (SAML assertion, WS-Trust STS SAML token, or WS-Federation JSON Web Token).
- User attributes that may be sent using the SAML Attribute Query profile (if that profile is used).
- The protocol, profiles, and bindings of the connection, including detailed security specifications (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 (see).

If your agreement includes sending assertions containing attribute values from local data stores, you must define the required data stores (see *Managing data stores* on page 166).

Administrative interface

You manage connection settings using the **SP Connection** wizard, 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 screen, where you manage one or more settings. You may move to a sibling task using the **Next** or **Previous** button. If you are on a sub task, you may also move to its parent task using the **Done** button.

When creating a new connection, you may save your progress using the **Save Draft** button. Note that not all screens offer this option. When you reach the **Activation & Summary** screen, you must click **Save** to complete the new connection.

When editing an existing connection, you may make changes and then click **Save** to commit your changes. In order words, you are not required to step through all screen to reach the **Activation & Summary** screen before you can save your changes.

Note: The **Save** button is available on most screen. If a screen does not show a **Save** button, click **Next** or **Done** until you reach to a screen where you can use its **Save** button to commit your changes.

Access SP connections

About this task

The **Identity Provider** menu displays a list of the most-recently modified SP connections. You may create or import a connection. You may also edit a recently modified connection by clicking on its connection name.

To access the rest of the connections, click **Manage All** to open the **SP Connections** screen. The **SP Connections** screen displays 20 connections at a time. As needed, you can 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, and default virtual server ID; narrow by protocol and status; and perform various connection-related tasks.

Steps

- To edit a connection, select the connection by its name. For the setting you want to make a change, select the corresponding screen title and then follow the configuration wizard to complete the task.
- To create a connection, follow the **Connection** configuration wizard to create a new connection to your SP partner.
- To copy a connection, select **Copy** under **Action** and then follow the **Connection** 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.
- To export a connection, select **Export Connection** under **Action** 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 making changes to it.
- To import a connection, click **Import Connection** and 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 may modify the XML file to suit your needs. The XML file may also be imported to another PingFederate environment acting in the same federation role (IdP) at your site. Note that the source and the target must run the same version of PingFederate.
- To export metadata for any SAML Browser SSO connection, click **Export Metadata** under **Action** and then follow the on-screen instructions to complete the task.
- To update a SAML Browser SSO connection, select **Update with Metadata** under **Action** and then follow the on-screen instructions to complete the task.

  You may update a connection via a metadata XML file or a metadata URL.

  ⓘ **Important:** The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.
- To toggle the status of a connection, slide the toggle switch to enable or disable a connection.
- To remove a connection, select **Delete** under **Action**.
- 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.

Override option	Description
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** 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 the **Service Provider**# **IdP Connections** screen as well.

  For more information about this advanced setting and its impact, see *Configuring automatic connection validation* on page 250.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Resolve SP connection errors

About this task

PingFederate automatically validates configured connections before displaying them on the **Connections** screen. 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-screen instructions to modify the configuration, one connection at a time.

Import a connection

About this task

Use the **Import Connection** screen to import a connection.

Note: Prior to the import, you may modify the XML file to suit your needs. The XML file may also be imported to another PingFederate environment acting in the same federation role (IdP or SP) at your site. Note that the source and the target must run the same version of PingFederate.

Steps

1. Click **Import**.
2. On the **Import Connection** screen, browse to a connection XML file.
3. Select the **Allow Update** check box. When selected, if the connection already exists, it will be overwritten.
4. Click **Import** and **Done**.

Update a SAML connection using metadata

About this task

You can update an existing SAML connection using a metadata file or a metadata URL from your partner.

> **Note:** This manual update is independent from the optional and per-connection automatic update feature.

Steps

1. Go to the **Manage All# Connections** screen.
2. Click **Update with Metadata** under **Action** for the applicable SAML connection.
3. Refer to the following steps to import or update metadata. Instructions vary depending on the medium of the metadata.

Metadata medium	Steps
A metadata file	**a.** On the **Import Metadata** screen, select the **File** option. **b.** 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** screen, and then click **Next**. **c.** On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **d.** Click **Save**.
A metadata URL	**a.** On the **Import Metadata** screen, select the **URL** option. **b.** 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**. **c.** Click **Load Metadata**. **Note:** If there is a digital signature error, click **Manage Partner Metadata URLs** to resolve the issue. **d.** Click **Save**.

4. On the **Connections** screen, click **Save**.

Results

> **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 screens do not include this information; only relative paths are shown.

Example

An 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 IdP can update the SP connection using the metadata immediately.

1. Access the **Identity Provider# Manage All# Connections** screen.
2. Click **Update with Metadata** under **Action** for the applicable SAML connection.
3. Follow the workflow to complete the task.

Choose an SP connection template

About this task

On the **Connection Template** screen (shown only for a new connection), you can choose a quick-connection template if your installation includes an optional PingFederate SaaS Connector.

> **Tip:** When you select a connection template, many connection settings are configured for you automatically.

Steps

- To configure a connection without a template, click **Next**.
- To use a template, select that option, then choose the template and enter additional information as required.

  > **Note:** Once you click **Next**, you cannot return to this screen and make a different selection. If you intended to use a different template or no template, you must create a new connection.

Choose an SP connection type

About this task

If you are not using a connection template (which pre-configures browser-based SSO), indicate on the Connection Type screen 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. Note that each connection must use a unique (partner) connection ID.

Steps

- To configure a connection for secure browser-based SSO, select the **Browser SSO Profiles** check box.

  If you have selected multiple protocols on the **System**# **Protocol Settings**# **Roles & Protocols** screen and 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, namely **SAML 1.1** or **JWT** (JSON Web Token).

  > **Tip:** If you are creating a WS-Federation connection to Microsoft Windows Azure Pack, select JWT as the token type.
- To configure an STS connection, select the **WS-Trust STS** check box.

  The **WS-Trust STS** option is only available after you enable the **WS-Trust** role on the **System**# **Protocol Settings**# **Roles & Protocols** screen.

- To configure a connection for outbound provisioning, make that selection and then select the provisioning type from the list.

  The **Outbound Provisioning** option is only available after you enable the **Outbound Provisioning** protocol on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
- 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.

Choose SP connection options

About this task

On the **Connection Options** screen (shown only for browser-based SSO connections), you can enable browser-based SSO, Attribute Query, or both for the current connection. For browser-based SSO, you may also enable IdP Discovery.

Steps

- To create a connection for browser-based SSO, select the **Browser SSO** check box.
- To enable IdP Discovery for this connection, make that selection (after selecting the **Browser SSO** check box).

  Note that the **IdP Discovery** check box is only available if IdP Discovery is configured (see *Configuring standard IdP Discovery* on page 139).
- To create a connection to facilitate the SAML 2.0 Attribute Query profile, select the **Attribute Query** check box (see *Attribute Query and XASP* on page 45).

Import SP 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.

The reload frequency is configurable through the **Reload Delay** field on the **System**# **Metadata Settings**# **Metadata Lifetime** screen. The default reload frequency is daily.

Although optional, it is recommended that you turn on notifications for SAML metadata update events on the **System**# **Runtime Notifications** screen.

Note: If the metadata contains changes that require additional configuration, the notification message also provides a list of the applicable items.

After the connection is created, you can add, remove, or change the metadata URL associated with the connection in the **Import Metadata** screen. In addition, you can also toggle the **Enable Automatic Reloading** option 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 (see www.incommon.org/participants).

Steps

Refer to the following steps to import or update metadata. Instructions vary depending on the medium of the metadata.

Metadata medium	Steps
A metadata file	**a.** On the **Import Metadata** screen, select the **File** option. **b.** 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** screen, and then click **Next**. **c.** On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **d.** Click **Next**.
A metadata URL	**a.** On the **Import Metadata** screen, select the **URL** option. **b.** 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**. **c.** Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. **d.** 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. **e.** Click **Next**.

Identify the SP

About this task

On the **General Info** screen, 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

Provide the basic information to identify your partner.

Refer to 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 via a SAML metadata file. For a WS-Federation connection, this is your partner's Realm. For an STS-only connection, this ID can be 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 on the **System**# **Protocol Settings**# **Federation Info** screen, enter a virtual server ID in this field and click **Add**. Enter additional virtual server IDs as needed.
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.
Application Name	The name of the application, accessible through the IdP Adapter interface (`IdpAuthenticationAdapterV2`) in the PingFederate Java SDK. (Note that 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.

Populate extended property values

Steps

Optional: On the **Extended Properties** screen, add, remove, or update one or more values for any extended properties defined on the **System**# **Extended Properties** screen.

Applicable and shown only if one or more extended properties are defined on the **System**# **Extended Properties** screen.

Extended property values can serve as metadata. They can also help drive authentication requirements. To learn more, see .

Configure IdP Browser SSO

Browser-based SSO (also known as Browser SSO) relies on a user's web browser and HTTP requests to broker identity-federation messaging (in XML or JWT) between an IdP and an SP (in contrast to WS-Trust STS messaging, which is typically application-driven across the back channel and does not require browser mediation).

To continue, click **Configure Browser SSO**.

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 (see *Federation roles* on page 30). 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

- *Choose SAML 2.0 profiles* on page 507
- *Set an SSO token lifetime* on page 507
- *Configure SSO token creation* on page 508
  - *Choose an identity mapping method* on page 508
  - *Set up an attribute contract* on page 510
  - *Manage authentication source mappings* on page 513
- *Configure protocol settings* on page 523
  - *Set Assertion Consumer Service URLs (SAML)* on page 523
  - *Specify SLO service URLs (SAML 2.0)* on page 526
  - *Choose allowable SAML bindings (SAML 2.0)* on page 527
  - *Set an artifact lifetime (SAML)* on page 527
  - *Specify artifact resolver locations (SAML 2.0)* on page 528
  - *Define signature policy (SAML)* on page 528
  - *Configure XML encryption policy (SAML 2.0)* on page 529

SAML 1.x configuration steps

- *Set an SSO token lifetime* on page 507
- *Configure SSO token creation* on page 508
  - *Choose an identity mapping method* on page 508
  - *Set up an attribute contract* on page 510
  - *Manage authentication source mappings* on page 513
- *Configure protocol settings* on page 523
  - *Set Assertion Consumer Service URLs (SAML)* on page 523
  - *Set a default target URL (SAML 1.x)* on page 524
  - *Set an artifact lifetime (SAML)* on page 527
  - *Define signature policy (SAML)* on page 528

WS-Federation configuration steps

- *Set an SSO token lifetime* on page 507
- *Configure SSO token creation* on page 508
    - *Choose an identity mapping method* on page 508
    - *Set up an attribute contract* on page 510
    - *Manage authentication source mappings* on page 513
- *Configure protocol settings* on page 523
    - *Define a service URL (WS-Federation)* on page 525

After configuring SSO settings, you will normally need to configure authentication credentials, the range of which depends on your SSO selections (see *Configure credentials* on page 535). Also, other configuration tasks may remain to be configured for new or modified connections, depending on the selected options on the **Connection Options** screen.

Choose SAML 2.0 profiles

About this task

On the **SAML Profiles** screen, select one or more SAML 2.0 profiles.

Tip: A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use (in contrast to SAML binding, which is the transport protocol of SAML messages).

Note that the **SAML Profiles** screen is not shown for SAML 1.x connections because IdP SSO is assumed, SLO profiles are not supported, and the server supports the "destination-first" (SP-initiated) profile SSO automatically. This screen is also not presented for WS-Federation connections because profile selection is not required.

For SAML 2.0, PingFederate supports all IdP- and SP-initiated SSO and SLO profiles. (For information on typical SSO and SLO profile configurations, including illustrations, see *SAML 2.0 profiles* on page 36.)

Steps

- Select the applicable profile (or profiles) based on your partner agreement.

  You must always select at least one SSO profile.

Set an SSO token lifetime

About this task

Identity-federation standards require a window of time during which an SSO token is considered valid. Each SSO token has an issuance time-stamp element and elements indicating the allowable lifetime of the SSO token (in minutes) before and after the issuance time stamp.

Steps

- 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.

Configure SSO token creation

About this task

As an IdP, you must specify how PingFederate obtains user-authentication information and use it to create SSO tokens appropriate for your SP partner, including additional user attributes as needed.

If you are a federation hub, bridging a service provider to one or more identity providers, you may associate one or more authentication policy contracts to the SP connection (see for more information).

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

- To continue, click **Configure ...**.

Choose an identity mapping method

About this task

On the **Identity Mapping** screen, you choose the type of name identifier (*Name ID*) your partner requires. Your selection may affect the way that the 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 which protocol you are using, namely SAML or WS-Federation.

Steps

- Refer to subsequent topics for configuration steps.

  Note that the **Identity Mapping** screen is not applicable to connections using the WS-Federation protocol in conjunction with 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.

Select a SAML Name ID type

About this task

The choice you make on the **Identity Mapping** screen affects how your SP partner makes use of account mapping or account linking.

If your SP is using account linking, establishing an attribute contract is not required. However, depending on your agreement, you may 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 (see).

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.

Steps

Select the type of name identifier that you and your SP have agreed to use.

Option	Description
Standard	Select the **Standard** option 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 the **Pseudonym** option 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 identifier may also be used by the SP to make a persistent association between the user and a specific local account (account linking). Select the **Include attributes ...** check box if you want to set up an attribute contract to use in conjunction with an opaque identifier.
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. A typical application for this selection might be, for example, when an SP provides generalized group accounts based on organizational rather than individual identity. Select the **Include attributes ...** check box if you want to set up an attribute contract to use in conjunction with an opaque identifier.

Select a WS-Federation Name ID type

About this task

On the **Identity Mapping** screen, indicate to the nature of the subject identifier by selecting one the three Name ID types for WS-Federation. Your selection may affect the way that the SP looks up and associates your users to their local accounts.

Note that the **Identity Mapping** screen is not applicable to connections using the WS-Federation protocol in conjunction with 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

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 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 data store.
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 data store.
Common Name	This selection provides for anonymous SSO to your SP, generally using a hard-coded generalized logon. Make this selection if your partner agreement involves a many-to-one use case; for instance, if the SP has a group account set up for all users in a particular domain.

Set up an attribute contract

About this task

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in the SSO tokens for this connection (see). You identify these attributes on the **Attribute Contract** screen.

Establishing an attribute contract is required for WS-Federation connections. For SAML connections, attribute contracts are optional if you are sending either pseudonym or a transient identifiers to the partners. When establishing an attribute contract, you may 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** screen.

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** screen; otherwise the **Attribute Contract** screen 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** screen; otherwise the **Attribute Contract** screen is not shown.	Assumed and cannot be added as an additional attribute.	At least one is required. There is no attribute name format.

Protocol	Identity mapping	Attribute contract	SAML_SUBJECT	Additional attributes
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 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 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.

Steps

1. Optional: Select a different name format for the built-in subject identifier, **SAML_SUBJECT**.

   Applicable if you and the SP have agreed to a specific format (see *Attribute contracts* on page 121).

   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 of PingFederate is required to activate any changes made to this file.)

   (Other conditions described in the table apply.)
2. Extend the contract with additional attributes.

   (Conditions described in the table apply.)

   a. 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 IdP application.

      The value of this attribute can then be mapped later on the **Attribute Contract Fulfillment** screen (see *Configure contract fulfillment for IdP Browser SSO* on page 516). Note that 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:

- `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

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 www.incommon.org/participants), the attribute contract may 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 (tools.ietf.org/html/rfc4524) and RFC4519 (tools.ietf.org/html/rfc4519). The following table describes a subset of the OIDs (object IDs) referenced by the most common attributes used by InCommon participants.

OID value	Description
0.9.2342.19200300.100.1.3	mail
1.3.6.1.4.1.5923.1.1.1.6	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, refer to 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.

b. Select an attribute name format from the list.

   Applicable if you and the SP have agreed to a specific format (see *Attribute contracts* on page 121).

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 of PingFederate is required to activate any changes made to this file.)

c. Click **Add**.
d. Repeat until all desired attributes are defined.

Results

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.

Manage authentication source mappings

About this task

IdP adapters are responsible for handling user authentication as part of an SSO operation. A configured adapter in PingFederate is known as an adapter instance.

In a basic scenario, you map an IdP adapter instance to an SP connection on the **Authentication Source Mapping** screen 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 may map multiple APCs to an SP connection (because your policies use multiple APCs), the same APC to multiple SP connections (because you want to reuse authentication policies on multiple connections), or a combination of them.

Tip: To learn more about authentication policies and contracts, see *Authentication policies* on page 324.

Furthermore, you can 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: To learn more about federation hub, see *Federation hub use cases* on page 127.

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 have the option to leave the decision to the users or create authentication policies to mandate authentication requirements.) Because each adapter instance or APC may return different user attributes, each mapping must define how the attribute contract is fulfilled in its mapping configuration.

Steps

- To map an IdP adapter instance, click **Map New Adapter Instance**.
- To map an APC, click **Map New Authentication Policy**.
- To edit the mapping configuration of an IdP adapter instance or APC, open it by clicking on its name, select the setting that you want to reconfigure, and complete the change.
- To remove an IdP 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 authentication sources, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Results

When authentication sources (IdP adapter instances or connection mapping contracts) are restricted to certain virtual server IDs, the allowed IDs are displayed under **Virtual Server IDs**.

Select an authentication source

About this task

The first task of the mapping configuration is to map an adapter instance or an authentication policy contract to your connection.

Steps

- To map an adapter instance, follow these steps:
  a. Select an adapter instance from the list.

     If you do not see the desired adapter instance, click **Manage Adapter Instances** to create a new instance of any deployed adapter.
  b. Select the **Override Instance Settings** check box if you want to customize one or more adapter settings for this connection alone.

     When selected, the administrative console adds a new set of sub tasks (the **Override Instance** screen 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 plug-in configurations* on page 119.
- To map an APC, select an adapter instance from the list.

  If you do not see the desired APC, click **Manage Authentication Policy Contracts** to create a new policy contract.

Results

If you are editing a currently mapped adapter 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 adapter settings specifically for this connection.

If you are editing a currently mapped APC, no changes can be made on this screen.

Override an IdP adapter instance

About this task

On the **Override Instance** screen, 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 settings screens, select the **Override** check box, make your changes, and then click **Next**. When you are finished, click **Done** to continue with the rest of the mapping configuration.

  ⓘ **Note:** The override setting screens are functionally identical to those used for creating a new adapter instance (see).

Results

If you are editing a currently mapped adapter instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You may also remove all overridden settings on a per-screen basis by clearing the **Override** check box near the top of the screen.

Restrict an authentication source to certain virtual server IDs

About this task

When you multiplex one connection for multiple environments (see), you can enforce authentication requirements by restricting an authentication source to certain virtual server IDs on the **Virtual Server IDs** screen. By default, no restriction is imposed.

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.

Results

If you are editing a currently mapped adapter instance or APC, you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Select an attribute mapping method

About this task

On the **Mapping Method** screen, you select if and how PingFederate should query local data stores to help fulfill the attribute contract in conjunction with attribute values from the authentication source.

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, determine whether local data stores can supply it.

> **Tip:** Alternatively, you may configure data store queries as part of the fulfillment configuration for the applicable IdP adapter contract or authentication policy contract. If so, you do not need to set up data store query on the connection level.
>
> For more information, see *Define the IdP adapter contract* on page 486 or *Applying policy contracts or identity profiles to authentication policies* on page 349.

Steps

- Select the **Retrieve additional attributes from multiple data stores using one mapping** option if you want to configure one or more data stores to look up attribute for a single mapping.
- Select the **Retrieve additional attributes from a data store—includes options to use alternate data stores and/or a failsafe mapping** option if you want to define alternate data stores to look up attribute and a failsafe mapping configuration.

  > **Note:** When selected, the token authorization framework (through issuance criteria) does not apply. For more information, see *About token authorization* on page 124 and *Attribute mapping with multiple data sources* on page 870.
- Select the **Use only ...** option if you do *not* require connection-level data store query.

Results

If you are editing a currently mapped adapter instance or APC, you can change the mapping method, which may require additional configuration changes in subsequent tasks.

Configure attribute sources and user lookup

About this task

Attribute sources are specific data store or directory locations containing information that may be needed for the attribute contract. You can use more than one attribute source when mapping values to the attribute contract. The order matters and affects the queries differently based on the selection made on the **Mapping Method** screen.

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—includes options to use alternate data stores and/ or a failsafe mapping

As soon as a query succeeds, PingFederate moves on to the next task (contract fulfillment); therefore, you should prioritize the attribute sources.

Steps

- Click **Add Attribute Source** and then follow a series of sub tasks to complete the configuration.

  For step-by-step instructions, see *Choose a data store* on page 872.

  Repeat to add additional attribute sources.

Results

If you are editing a currently mapped adapter instance or APC, you can add, remove, or reorder attribute sources, which may require additional configuration changes in subsequent tasks.

Configure contract fulfillment for IdP Browser SSO

About this task

On the **Attribute Contract Fulfillment** screen, map values to the attributes defined for the contract. These are the values that will be included in the SSO tokens sent to the SP.

If you are bridging one or more identity providers to a service provider, map values to an authentication policy contract (see *Federation hub use cases* on page 127).

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

Steps

- For each attribute, select a source from the list and then choose or enter a value.
  - **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 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 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 especially 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 .

> **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values (see **Expression**).

  - **LDAP**, **JDBC**, or **Other**

    When selected, the **Value** list is populated 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; for example, 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 (see **Text**).

Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .

- **Text**

  When selected, the text you enter is mapped to the value of the attribute in the SSO 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 this syntax:

  `${ds.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:

  `${ds.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.

All attributes must be mapped.

Results

If you are editing a currently mapped adapter instance or APC, you can update the mapping configuration, which may require additional configuration changes in subsequent tasks.

Configure default contract fulfillment for IdP Browser SSO

About this task

If you have chosen the failsafe option on the ***Mapping Method*** screen and the **Send user to SP using default list of attributes** option on the ***Failsafe Attribute Source*** screen, define the default values that should be sent in the SSO tokens to the SP on the **Attribute Contract Fulfillment** screen.

Steps

- For each attribute, select a source from the list and then choose or enter a value.
  - **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 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 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 especially 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 .

    Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values (see **Expression**).
  - **Expression** (when enabled)

    This option provides more complex mapping capabilities; for example, 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 (see **Text**).

    Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
  - **Text**

    When selected, the text you enter is mapped to the value of the attribute in the SSO tokens at runtime. You can mix text with references to any of the values from the authentication source using the `${attribute}` syntax.

    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).

  All attributes must be mapped.

Results

If you are editing a currently mapped adapter instance or APC, you can update the mapping configuration, which may require additional configuration changes in subsequent tasks.

Define issuance criteria for IdP Browser SSO

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

Note: The token authorization feature does not apply if you have chosen the failsafe option on the **Mapping Method** screen (and the **Issuance Criteria** screen is not shown).

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that 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 (see), 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 (see).

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See the following table for more information.

Source	Description
Adapter or Authentication Policy Contract	Select to evaluate attributes from an IdP adapter instance or an authentication policy contract.
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 data store (if configured)	Select to evaluate attributes returned from a data source.

Source	Description
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   Error results are handled differently for IdP-initiated 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 `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, display an error message or e-mail an administrator.

If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the authentication source mapping

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the SSO token creation summary

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure protocol settings

About this task

The **Protocol Settings** screen provides the launching point for configuring partner endpoints, message customizations, and other protocol-specific settings for browser-based SSO connections.

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

- To continue, click **Configure Protocol Settings**.

Set Assertion Consumer Service URLs (SAML)

About this task

The assertion consumer service (ACS) endpoint is a location to which the SSO tokens are sent, according to partner requirements. ACS is applicable to all SAML versions and both the IdP- and SP-initiated SSO profiles.

On the **Assertion Consumer Service URL** screen, select the applicable SAML binding and enter the corresponding ACS endpoint URL.

Note: The SP may request that the SAML assertion be sent to one of several URLs, via 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 may 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. Configure one or more SAML ACS endpoints.
   a. Select a SAML binding from the list; for example, **POST**.
   b. Enter the ACS endpoint URL to the **Endpoint URL** field.

      You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.
   c. Make the selection 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 may reset the default selection when you add another ACS endpoint.
   d. 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.
   e. Click **Add**.
   f. Optional: Repeat to add additional ACS endpoints.
2. Optional: Customize messages using OGNL expressions.

   Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see *Attribute mapping expressions* on page 859.

   a. Click **Show Advanced Customizations**.
   b. Select a message type from the list.
   c. Enter an OGNL expression to fulfill your use case.

      Note: For more information about **Message Type**, available variables, and sample OGNL expressions, see *Customize assertions and authentication requests* on page 864 .
   d. Click **Add**.
   e. Optional: Repeat to add another message customization.

Results

If you are editing an existing connection, you can reconfigure any items, which may require additional configuration changes in subsequent tasks. You must always configure at least one ACS endpoint.

Set a default target URL (SAML 1.x)

About this task

SAML 1.x SP connections requires that a default target URL be specified for a scenario where the IdP application does not include one in its SSO request. This default URL represents the destination on the SP where the user will be directed.

Steps

- Enter default destination in the **Default Target URL** field.

Results

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.

Define a service URL (WS-Federation)

About this task

On the **Service URL** screen, enter the WS-Federation protocol endpoint of your SP partner where PingFederate sends SSO tokens and SLO cleanup messages. The SSO tokens are transmitted within an RSTR (Request for Security Token Response) message in response to a request for authentication from the SP. SLO cleanup messages are sent to your partner when PingFederate (the IdP) receives a user's SLO request. Such 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 screen. If the option to validate wreply for SLO is enabled, these additional domains and paths will also be taken into consideration as well (see *Manage partner redirect validation* on page 319).

Some federation use cases may 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. Enter the WS-Federation protocol endpoint at the SP site in the **Endpoint URL** field.

   You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.
2. Optional: Specify additional allowed domains and paths.
   a. 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.
   b. Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

      Enter a value without the protocol; for example: `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.
   c. Optional: Enter the exact path of this resource under **Valid Path**.

      Starts 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 may allow specific query parameter (or 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.

d. Optional: 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.

e. 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.

f. Repeat these steps to define multiple expected resources.

   Note that 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.

Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see *Attribute mapping expressions* on page 859.

a. Click **Show Advanced Customizations**.
b. Select a message type from the list.
c. Enter an OGNL expression to fulfill your use case.

> **Note:** For more information about **Message Type**, available variables, and sample OGNL expressions, see *Customize assertions and authentication requests* on page 864 .

d. Click **Add**.
e. Optional: Repeat to add another message customization.

Results

If you are editing an existing connection, you can reconfigure any items, which may require additional configuration changes in subsequent tasks.

Specify SLO service URLs (SAML 2.0)

About this task

On the **SLO Service URLs** screen, you associate bindings to the endpoints where your SP receives logout requests when SLO is initiated at your site and where PingFederate sends SLO responses when it receives SLO requests from the SP.

This step applies only to SAML 2.0 connections when either SLO profile is selected on the **SAML Profiles** screen.

Steps

1. Select a SAML binding from the list; for example, **POST**.

2. Enter the SLO endpoint URL to the **Endpoint URL** field.

You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.

3. Optional: Enter an URL to the **Response URL** field.

   When specified, it 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 may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.
4. Click **Add**.
5. Optional: Repeat to add additional SLO endpoints.

Results

If you are editing an existing connection, you can reconfigure the SLO endpoints, which may require additional configuration changes in subsequent tasks.

Choose allowable SAML bindings (SAML 2.0)

About this task

On the **Allowable SAML Bindings** screen, you select the one or more bindings that your SP partner can use to send SAML authentication requests or SLO messages.

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** screen.

Steps

- Select only the applicable SAML binding (or bindings) based on your partner agreement.

  If you have specified an ACS or SLO endpoint using the artifact (outbound) binding, you must including SOAP as one of the allowable (inbound) binding.

Results

If you are editing an existing connection, you can reconfigure the allowable bindings, which may require additional configuration changes in subsequent tasks.

Set an artifact lifetime (SAML)

About this task

When PingFederate sends an artifact to your 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** screen, specify the expiry information in seconds.

You can change the default value per your requirements, if needed. Also consider synchronizing clocks between your server and your partner's SAML gateway server. If clocks are not synchronized, you might need to set the artifact lifetime to a higher value.

Steps

- Optional: Override the default value of the **Artifact Lifetime** field.

  The default value is 60 (seconds).

Results

If you are editing an existing connection, you can update the artifact lifetime.

Specify artifact resolver locations (SAML 2.0)

About this task

When the artifact binding is enabled as one of the allowable bindings on the **Allowable SAML Bindings** screen, you must provide at least one artifact resolution service (ARS) endpoint on the **Artifact Resolver Locations** screen. This is the location where PingFederate sends back-channel requests to resolve artifacts received from the SP.

Steps

1. Enter the ARS endpoint URL to the **URL** field.

   You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.
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, 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.

> **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.

Results

If you are editing an existing connection, you may reconfigure any ARS endpoints.

Define signature policy (SAML)

About this task

The **Signature Policy** screen provides options controlling how digital signatures are used for SAML messages. The choices made on this screen depend on your partner agreement (see).

SAML 2.0

Digital signing is required for SAML response messages sent from the IdP via the POST or redirect binding. Based on the SAML specifications, PingFederate provides three options:

- Select the **Always Sign Assertion** check box alone to always sign the assertion portion inside the SAML response message.
- Select the **Sign Response As Required** check box alone to sign the SAML response message per the SAML specifications. (This is the default selection.)
- Select *both* check boxes 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 SP may also be signed to enforce security. This scenario applies only when the SP-initiated SSO profile is enabled on the **SAML Profiles** screen.

- Select the **Require Authn Requests to be Signed ...** check box to enforce this digital signature requirement.

SAML 1.x

For SAML 1.0 and SAML 1.1, the assertion portion inside the SAML response message can be digitally signed.

- Select the **Always Sign Assertion** check box to always sign the assertion portion inside the SAML response message.

Steps

- To continue, select the option (or options) based on your partner agreement.

Results

If you are editing an existing connection, you can reconfigure the digital signature policy, which may require additional configuration changes in subsequent tasks.

Configure XML encryption policy (SAML 2.0)

About this task

For SAML 2.0 configurations, in addition to using signed assertions to ensure authenticity, you and your partner may also agree to encrypt all or part of an assertion to improve privacy. If so, you can configure these settings on the **Encryption Policy** screen. There are three options; each option provides different capabilities.

Option	**Name identifier (SAML_SUBJEC**	**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

- To continue, select the option (and options) based on your partner agreement.

Results

If you are editing an existing connection, you can reconfigure the XML encryption policy, which may require additional configuration changes in subsequent tasks.

Review protocol settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review browser-based SSO settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure the Attribute Query profile

About this task

At the Attribute Query step you configure your connection to respond to requests for user attributes from your partner SP, if you have chosen this option (see). Attribute queries are not dependent on single sign-on but may be used independently or in conjunction with Browser SSO or provisioning to provide flexibility in how a user authenticates with SP applications (see).

Steps

- To continue, click **Configure Attribute Query Profile**.

Define retrievable attributes

About this task

On the **Attribute Query# Retrievable Attributes** screen, you specify the user attributes you and your partner have agreed to allow in an attribute query transaction. Note that the SP may 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

- To add an attribute, enter the attribute name in the text box and then click **Add**.

  Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.
- To modify an attribute name, follow these steps:
  a. Click **Edit** under **Action** for the attribute.
  b. 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.

Configure attribute lookup

About this task

Attribute sources are specific data store or directory locations containing information that may be returned to the SP in response to an attribute request. This optional configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

Steps

- To configure an attribute source, click **Add Attribute Source** and complete the setup steps (see *Choose a data store for Attribute Query* on page 531).
- 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 may need to modify dependent data in subsequent steps, as indicated.

Choose a data store for Attribute Query

About this task

The process of configuring PingFederate to look up attributes in a data store for attribute-query responses is similar to that used for SSO Attribute Sources and User Lookup. On the **Data Store** screen, choose a data store instance for PingFederate to look up attributes.

Steps

1. Enter a description (and ID if prompted) for the data store.
2. Select a data store instance from the **Active Data Store** list.

> **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.

3. Depending on the data store type, the rest of the setup varies as follows:

Data store type	Required tasks
JDBC	▪ ▪
LDAP	▪ ▪ (optional) ▪
Other	▪

> **Important:** When attribute queries are sent using XASP, use the variable `${SubjectDN}`—rather than `${SAML_SUBJECT}`—to retrieve the subject identifier. You may also use any of these DN-parsing variables: `${CN}`, `${OU}`, `${O}`, `${L}`, `${S}`, `${C}`, and `${DC}`.
>
> If more than one value exists for any of the parsing variables, then they are enumerated. For example, if the Subject DN is:
>
> `cn=John Smith,ou=service,ou=employee`
>
> then you could use any of these elements in your filter qualifier:
>
> `${SubjectDN}=cn=John Smith,ou=service,ou=employee`
>
> `${ou}=service`
>
> `${ou1}=employee`
>
> For more information about XASP, see *Attribute Query and XASP*.

Configure contract 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.

You map attributes on the Attribute Mapping Fulfillment screen.

Map each attribute into the assertion from one of these Sources:

- **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 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 especially 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 .

  > **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values (see **Expression**).

- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.

- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 859). All of the variables available for text entries (see below) are also available for expressions.

- Text

  This can be text only, or you can mix text with references to any of the values from your user-data store using this syntax:

  ```
  ${ds.attr-source-id.attribute}
  ```

  where `attr-source-id` is the **Attribute Source ID** value (see *Choose a data store for Attribute Query* on page 531) and `attribute` is any of the data store 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.

Define issuance criteria for Attribute Query

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.

3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Specify security policy

About this task

This screen allows you to specify the digital signing and encryption policy to which you and your partner have agreed. These selections will trigger requirements for setting up Credentials (see).

Steps

- Check or clear the check boxes and click **Next** or **Done**.

Review the Attribute Query configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.
- To discard your changes, click **Cancel**.

Configure credentials

About this task

The **Credentials** screen 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**.

  Refer to subsequent topics for configuration steps.

Configure back-channel authentication (SAML)

About this task

Depending on your Browser SSO use cases, the administrative console prompts you to configure authentication requirements for inbound messages, outbound messages, or both. Refer to the following table for more information.

Use case	Back-channel authentication requirements	Back-channel messages
A connection is configured with a SAML ACS endpoint that uses the artifact binding on the **Protocol Settings# Assertion Consumer Service URL** screen.	Inbound	Artifact resolution requests
A connection is configured with a SAML 2.0 SLO endpoint that uses the artifact binding on the **Protocol Settings# SLO Service URLs** screen	Inbound	Artifact resolution requests SOAP messages

Use case	Back-channel authentication requirements	Back-channel messages
A connection is configured with a SAML 2.0 SLO endpoint that uses the SOAP binding on the **Protocol Settings# SLO Service URLs** screen	Outbound	SOAP SLO messages
The SAML 2.0 Artifact binding is enabled on the **Protocol Settings# Allowable SAML Bindings** screen	Outbound	Outbound artifact resolution requests
The SOAP binding is enabled on the **Protocol Settings# Allowable SAML Bindings** screen	Inbound	Inbound SOAP messages
The SAML 2.0 Attribute Query profile is enabled on the **Connection Options** screen	Inbound	Inbound Attribute Query requests

Steps

- Refer to subsequent topics for configuration steps.

Configure authentication requirements for outbound messages

Steps

1. On the **Back-Channel Authentication** screen, click **Configure** to the right of the list of messages under **Send to your partner**.
2. On the **Outbound SOAP Authentication Type** screen, 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)** screen.

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** screen. If you have not yet created or imported the client certificate, click **Manage Certificates** to do so (see).

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 a subsequent screen, **Digital Signature Settings**.

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 may be used in any combination or independently.

3. On the **Summary** screen, review your configuration and perform one of the following tasks:

   Amend your configuration

   Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

   Discard your changes

   Click **Cancel**.

Configure authentication requirements for inbound messages

Steps

1. On the **Back-Channel Authentication** screen, click **Configure** to the right of the list of messages under **Received from your partner**.
2. On the **Inbound Authentication Type** screen, 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)** screen.

   > ⓘ **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 screens (see next step).

   Digital Signature (Browser SSO profile only)

   You select a signing certificate on a subsequent screen, **Signature Verification Settings**.

   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 may 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 may 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** screen.

Anchored

The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you may 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 **Security**# **Trusted CAs** screen.

Unanchored

The partner certificate is self-signed or you want to trust a specified certificate.

Note: When anchored certificates are used between partners, certificates may be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated.

(For more information, see .)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: **a.** Enter the Subject DN of the certificate. **b.** (Optional) 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** screen, 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 (see).

4. On the **Summary** screen, review your configuration and perform one of the following tasks:

Amend your configuration

Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Configure digital signature settings

About this task

Digital signing is required for browser-based SSO tokens and SLO messages sent via POST or redirect bindings. It is also required for WS-Trust STS SP connections (for the purpose of signing the outbound SAML security tokens).

On the **Digital Signature Settings** screen, select the certificate that you will use to sign the SSO tokens and SLO messages for this SP.

Note that, 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 the **Protocol Settings**# **Signature Policy** screen or the artifact resolution messages on the **Back-Channel Authentication**# **Outbound SOAP Authentication Type** screen.

If digital signing is not required, the **Digital Signature Settings** screen is not shown.

Steps

1. Select a signing certificate from the list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing certificates and decryption keys* on page 299).

   Note: For WS-Federation connections using JSON Web Tokens, only EC and RSA certificates are supported. Furthermore, 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.

2. 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 JSON Web Tokens.

   Select the **Include the raw key in the signature <KeyValue> element** check box if your partner agreement requires it.

3. Optional: 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.

Configure signature verification settings (SAML 2.0)

About this task

Depending on your partner agreement, digital signature processing may be required.

If you chose to require digital signatures on SAML 2.0 authentication requests on the **Protocol Settings**# **Signature Policy** screen or inbound messages on the **Back-Channel Authentication**# **Inbound Authentication Type** screen, you must configure the required certificate information that PingFederate can use to verify the signed messages.

The **Signature Verification Settings** is the launching point for this task. If digital signature verification is not required, the **Signature Verification Settings** screen is not shown.

Steps

1. On the **Signature Verification Settings** screen, click **Manage Signature Verification Settings**.
2. On the **Trust Model** screen, select a trust model on the **Certificate Verification Method** screen.

   Anchored

   The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you may 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 **Security# Trusted CAs** screen.

   Important: If you are using the redirect binding for 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 may 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* on page 117.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: **a.** Enter the Subject DN of the certificate or extract it from your SP partner's certificate if the certificate is stored on an accessible file system. **b.** (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** screen: **a.** Select a certificate from the list. If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so (see *Manage certificates from partners* on page 310). **b.** (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 situation 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** screen, review your configuration and perform one of the following tasks:

Amend your configuration

Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Select an encryption certificate

About this task

For browser-based SSO, if you chose to encrypt all or part of an SSO assertion on the **Protocol Settings# Encryption Policy** screen, you must identify the certificate that PingFederate can use to do so.

You must also select a certificate if your requirements include encrypting an assertion in response to an attribute query on the **Attribute Query# Security Policy** screen.

For WS-Trust STS, this configuration is also required if you have enabled the **Generate Key for SAML Holder of Key Subject Confirmation Method** or **Encrypt SAML 2.0 Assertion** option (or both options) on the **WS-Trust# Protocol Settings** screen.

If encryption is not required, the **Select XML Encryption Certificate** screen is not shown.

Steps

1. Optional: Select an option under **Block Encryption Algorithm**.

   Important: Due to the import restrictions of some countries, Oracle Server JRE (Java SE Runtime Environment) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see the Java 8 release notes from Oracle (www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html).

   For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide (docs.oracle.com/en/java/javase/11/migrate/).

   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 (www.w3.org/TR/xmlenc-core/).
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, we recommend upgrading such connections to use the newer algorithm **RSA-OAEP**.

   The default selection is **RSA-OAEP**.
3. Select a partner certificate from the list.

   If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so (see).

Select a decryption key (SAML 2.0)

About this task

When you chose to encrypt the name identifier (SAML_SUBJECT) on the **Protocol Settings# Encryption Policy** screen, you also have the option to allow the SP to encrypt the name identifier in its SLO requests (if the SP-initiated SSO profile is enabled for the connection). To enable this inbound encryption, you must specify at least one certificate on the **Select Decryption Keys** screen.

If decryption is not required, the **Select Decryption Keys** screen is not shown.

Steps

1. Select the primary XML decryption key from the list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see).
2. Optional: Select the secondary XML decryption key from the list.

Review credential settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure outbound provisioning

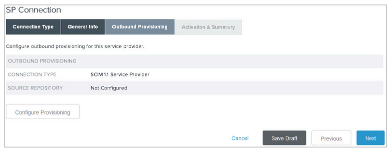
About this task

PingFederate's Outbound provisioning allows an IdP to create and maintain user accounts at standards-based partner sites using SCIM as well as select-proprietary provisioning partner sites that are protocol-enabled (see *Outbound provisioning for IdPs* on page 125).

Note: This configuration task is presented in the administrative console only when you enable the **Outbound Provisioning** protocol (see *Choose an SP connection type* on page 502).

Steps

- To continue, click **Configure Provisioning**.



SP Connection

Configure outbound provisioning for this service provider.

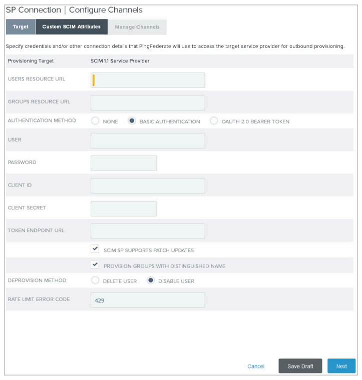
OUTBOUND PROVISIONING	
CONNECTION TYPE	SCIM 1.1 Service Provider
SOURCE REPOSITORY	Not Configured

Note: Screen illustrations in this section are presented in most cases for service providers using the SCIM protocol. However, the screens are comparable for SaaS vendors for whom provisioning is supported, and the information and instructions provided are the same unless otherwise noted.

Define a provisioning target

About this task

Information on the **Target** screen indicates 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 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 SCIM outbound provisioning and various 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, please refer to documentation in the add-on distribution package.

The following steps describe the fields required for the bundled PingFederate provisioning plug-in 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. Configure the rest of the 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**
User, and	Valid credentials to access the partner's endpoint.
Password	Required if **Basic Authentication** is the selected authentication method.
Client ID, Client Secret, and	Valid OAuth client credentials and token endpoint to access the partner's endpoint.
Token Endpoint URL	Required if **OAuth 2.0 Bearer Token** is the selected authentication method.
SCIM SP Supports Patch Updates	Clear this check box if the partner does *not* support PATCH updates. For information about PATCH, see the *SCIM specification* (www.simplecloud.info/specs/draft-scim-api-01.html#edit-resource-with-patch). This check box is selected by default.
Provision Groups with Distinguished Name	Select this check box to provision groups by supplying complete LDAP 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 (for example, 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 check box is selected by default.

Field	Description
Deprovision Method	Deprovisioning is triggered when previously provisioned users no longer meet the condition set in the **Manage Channels**# **Channel**# **Source Location** screen. 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. Rather their status is changed 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.

3. Click **Next**.

> **Note:** For some provisioning plug-ins (including the built-in SCIM outbound provisioner),when you first enter or change credentials and click **Next**, PingFederate immediately tests connectivity to the target.

Specify custom SCIM attributes

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** screen.

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 attributes

The 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.

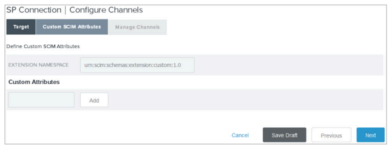
```
{
  "userName":"CBrown",
  "active":true,
  "schemas":[
    "urn:scim:schemas:core:1.0",
    "urn:scim:schemas:extension:custom:1.0"
  ],
  ...
  "urn:scim:schemas:extension:custom:1.0":{
    "supervisor":"JSmith",
    "territories":[
      "Montana",
      "Idaho",
      "Wyoming"
    ],
    "options":{
      "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. 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

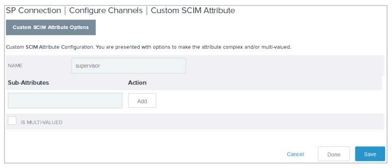
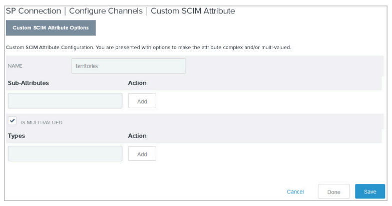
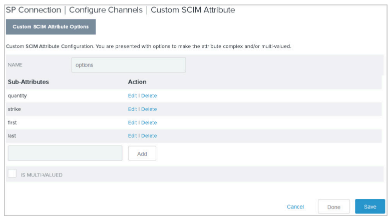
Tip: The default value is `urn:scim:schemas:extension:custom:1.0`. You can keep this value if your partner identifies custom attributes by this URI in its SCIM messages.

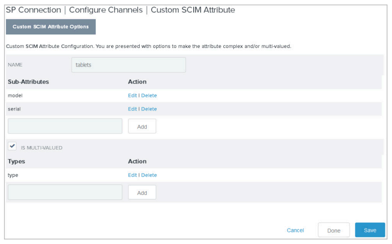
2. Enter an attribute name and click **Add** to add a custom attribute.

   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:

Task	Steps
Change the attribute name	**a.** Replace the current value in the **Name** field. **b.** 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: supervisor Sub-Attributes | Action IS MULTI-VALUED: ☐
Set the attribute as a simple multivalued attribute	**a.** Select the **Is Multivalued** check box. **b.** 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 IS MULTI-VALUED: ✓ Types | Action
Add sub-attributes to make the attribute a complex attribute	**a.** Enter a sub-attribute and click **Add**. (Repeat this step to add more sub-attributes as needed.) **b.** 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. **c.** 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: options Sub-Attributes | Action quantity | Edit | Delete strike | Edit | Delete first | Edit | Delete last | Edit | Delete IS MULTI-VALUED: ☐

Task	Steps
Add sub-attributes and set the attribute as a complex multivalued attribute	**a.** 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. **b.** Select the **Is Multivalued** check box. **c.** 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. **d.** 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

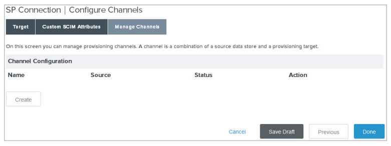
When you finish editing custom attributes, click **Next** in the **Custom SCIM Attributes** screen.

Manage 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; for example, if your organization has separate LDAP stores (or different nodes in the same store) for various user groups needing SSO access and provisioning to the same domain.

ⓘ **Tip:** There can be only one target domain for provisioning per connection. If your organization subscribes to multiple domains for which you need provisioning support, you need a separate SP connection for each domain.


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

On the **Manage Channels** screen, you can perform the following tasks:

Steps

- Click **Create** to add a new channel.

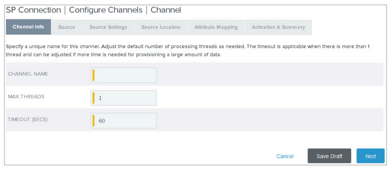
  Alternatively, you create a new channel by copying an existing channel (and making other requird changes).
- Select an existing channel to modify its settings.

- Use the **Delete** and **Undelete** workflow to remove or cancel the removal request of existing channel.

Specify channel information

About this task

In the **Channel Info** screen, 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.



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. Enter a channel name.

   If you are copying a channel, you must enter a new value in the **Channel Name** field.
2. Optional: Update the values for the **Max Threads** and **Timeout** fields.

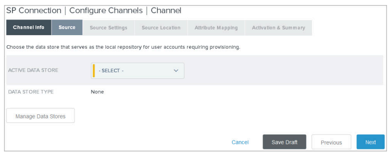
   The **Timeout** value applies only when the **Max Threads** value allows multiple threads.
3. Click **Next**.

Identify the source data store

About this task

PingFederate supports PingDirectory™ , Microsoft Active Directory, and the 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-data store is used to supply mapped values for each user attribute required by the 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

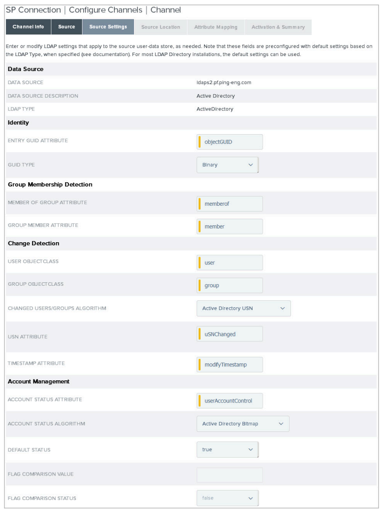
1. On the **Source** screen, choose the LDAP store to use for this channel.

   If the data store 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 data store.
2. Click **Next**.

Modify source settings

About this task

The **Source Settings** screen shows the default configuration of the data store selected on the **Source** screen, including settings used by the PingFederate provisioner to determine when user information is added, changed, or removed.



SP Connection | Configure Channels | Channel

Channel Info | Source | Source Settings | Source Location | Attribute Mapping | Activation & Summary

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 data store representing the user's Globally Unique Identifier.
GUID Type	Indicates whether the GUID is stored in binary or text format. Active Directory is always binary. Other LDAP stores most often use text.
Member of Group Attribute	A multivalued user attribute containing the DNs of the groups to which an entry belongs. This attribute does not apply to some LDAP servers, including the Oracle Directory Server. The attribute below is used instead. Active Directory uses both values to provide a two-way mapping between User and Group objects.
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.
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 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 point of view of the PingFederate provisioner but can be more time consuming on the LDAP side, particularly with the 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 the Oracle Directory Server.
USN Attribute	The name of the attribute used to store the update sequence number—applicable when the Active Directory algorithm is chosen above.

Field	Description
Timestamp Attribute	The name of the attribute used to store the timestamp on user records.
Account Status Attribute	The name of the attribute in which the user's account status (active or inactive) is stored, for example, Active Directory = `userAccountControl` and Oracle Directory Server = `nsaccountlock`.
Account Status Algorithm	The method by which PingFederate determines a user's account status. The values are: **Active Directory Bitmap** – For Active Directory, which uses a bitmap for each user entry. For more information about `userAccountControl` flags, see Microsoft's *knowledge base* (support.microsoft.com/kb/305144). **Flag**– For 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 (for example, `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, and the Oracle Directory Server, in most cases no changes are needed on this screen unless your data store uses a customized schema.

If you are using a different LDAP directory, you must supply the required information on this screen unless you have defined a template for the data store. (For more information, see the `sample.template.txt` in the `<pf_install>/pingfederate/server/default/conf/template/ldap-templates` directory.)

Steps

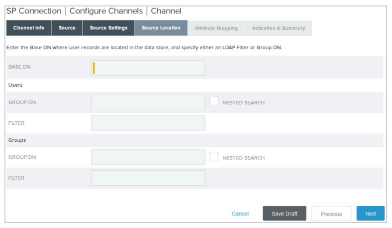
1. Modify the settings, as needed.
2. Click **Next**.

Results

Specify a source location

About this task

Indicate on the **Source Location** screen where PingFederate should look for user records in the data store. The same location may be used to retrieve user-group DNs for maintaining corresponding groups at the service provider.


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 have the options to provision users (and groups when applicable) based on group membership information or LDAP search results.

Note: Groups provisioning is supported for SCIM and the Google Apps Connector (version 2.0 and higher) but may not be supported for other SaaS Connectors. If not, the associated fields under **Groups** on the **Source Location** screen are inactive. Support for the feature may become available in future SaaS Connector releases; please refer to documentation in your add-on distribution package.

Steps

1. Enter the base DN where user records are stored in the **Base DN**.

   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, as described in the following table:

Object	Field description
Users	**Group DN** The distinguished name (DN) of a group in the user repository whose *member users* should be provisioned. (Optional) 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 Microsoft Active Directory or Oracle Directory Server is selected as the source user repository (see *Identify the source data store* on page 549). **Filter** An LDAP search filter that returns *user objects* representing the *users* that should be provisioned. For information about LDAP filters, refer to your LDAP documentation. Note that you may need to escape any special characters. **Important:** The **Group DN** field is ignored when a **Filter** field value is configured.

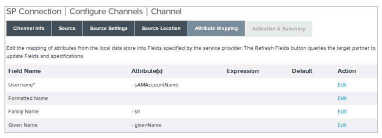
Object	Field description
Groups (when applicable)	**Group DN** The distinguished name (DN) of the group in the user repository that should be provisioned. (Optional) Select the **Nested Search** check box to include *groups* that are members of the specified group through nested group membership. Nested 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 Microsoft Active Directory or Oracle Directory Server is selected as the source user repository (see *Identify the source data store* on page 549). **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. Note that you may 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**.

Map attributes

About this task

The **Attribute Mapping** screen provides a means of managing how attributes from your user store are mapped to SCIM attributes in the core schema and custom attributes through a schema extension or to the provisioning fields supported for your organization's SaaS-customer account.



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 may make one or more optional core attributes mandatory. Refer to 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 screen, but only on the first pass through the screen flow. If you are using this screen to modify an existing mapping configuration, click **Refresh Fields** near the bottom of the screen to synchronize the list with the target if needed.

For each field, the **Specify Attribute Mapping** screen 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** near the of the screen 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. Click **Edit** under Action for a field.

   ⓘ **Tip:** If you have specified any custom attributes, they are listed at the end of the **Attribute Mapping** screen.

2. On the **Specify Attribute Mapping** screen, provide mapping details.
3. Repeat for each attribute shown in the **Field Name** column as needed.

   ⓘ **Tip:** For most fields, if you need to map more than one attribute from your data store into a single field at the target location, then you must use an OGNL expression to indicate how the attribute values are to be combined.

   The only exception is a field called LDAP Attributes Map, provided primarily to support SCIM attributes specific to PingOne® for Enterprise. This field may contain multiple attributes without using OGNL.

4. Click **Next**.

Specify mapping details

On the **Specify Attribute Mapping** screen, you define specific mapping information for each field required for provisioning (and for any optional fields, 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

1. **Optional:** Select the class containing a user-store attribute under **Root Object Class** that you want to map to the provisioning attribute shown under **Field Name**.

   ⓘ **Note:**

   For some fields, you may not need to map specific user attributes. If so, supply a value in the **Default Value** field instead—skip this step and go to step 5. You can also do both for certain attributes, as needed: that is, specify both LDAP attributes and a default value.

2. Select the source attribute from the class under **Attribute** and then click **Add Attribute**.
3. Repeat the previous steps to add additional applicable attributes, as needed, to use in a mapping expression.

   ⓘ **Important:**

   You must add an attribute for it to be used in an expression.

4. Enter or select a default value under **Value Definition** (optional, if one or more attributes is specified above).

   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 evaluating the expression.
5. If more than one attribute is used for mapping fields other than **LDAP Attributes Map**, enter an expression.

   > **Tip:**
   >
   > Click **Edit** to create and validate the expression.
6. **Optional:** Select one or more processing options, as defined below:

   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 value(s).

   Mask Log Values

   Determines whether sensitive information (for example, the `Password` attribute) will be masked in PingFederate log files.

   Upper Case, Lower Case, or None

   Transforms the attribute value(s) 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 (for example, 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.
7. Click **Done**.

Review channel settings

About this task

When you finish setting up a channel, you may choose to activate it immediately; or you can return to the **Activation & Summary** screen and activate the channel when needed. Note that the SP connection must also 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 SSO and SLO transactions may still occur (if an associated connection is active).

Steps

- To toggle the status, select **Active** or **Inactive**, and then 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* on page 138). The default is 60 seconds. Initial provisioning can consume considerable processing time, depending on the amount of data that needs to be transmitted; administrators may wish to plan accordingly.

- To modify channel settings, click the associated heading.

> ⓘ **Important:** Regardless of whether you choose to activate a new channel immediately or later, if you want to save the channel configuration, click **Save** on the **Activation & Summary** screen.

Review SP connection settings

About this task

When you finish creating or modifying a connection, you can review the connection settings and toggle the connection status on the **Activation & Summary** screen.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Results

> ⓘ **Important:** When creating a new connection, the default connection status is **Enabled** when you reach the **Activation & Summary** screen.
>
> Regardless of whether you choose to disable a new connection now or later, you must click **Save** on the **Activation & Summary** screen 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 may use to invoke SSO for the connection. For a list of supported parameters, see *View IdP application endpoints* on page 496.

SP affiliations

An 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 (see *Account linking* on page 120).

This may be of use when multiple service providers 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 IdP adapter instance for generating account links (see *Manage authentication source mappings* on page 513).

Manage SP affiliations

About this task

The **Identity Provider** menu displays a list of the most-recently modified SP affiliations. You may create a new SP affiliation or edit a recently modified SP affiliation by clicking on its name.

To access the rest of the SP affiliations, click **Manage All** to open the **SP Affiliations** screen.

Steps

- To create a new SP affiliation, click **Create New** under **SP Affiliations** on the **Identity Provider** menu.

  If you are on the **Identity Provider# SP Affiliations** screen, click **Create Affiliation**.
- To edit an SP affiliation, select the affiliation by its ID and follow configuration wizard to complete the task.
- To delete an SP affiliation, click **Delete** under **Action** for the SP affiliation.

Import affiliation metadata

About this task

An IdP may send a metadata file containing information that automatically specifies members of an SP affiliation for use in PingFederate.

Steps

- To import a metadata file, click **Choose File** to upload it.
- If you do not have a metadata file, click **Next**.

Enter affiliation information

Steps

- On the **Affiliation General Info** screen, provide the requested information, as described in the following table.

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 may serve as the Owner.

  If you imported a metadata file, this information is already supplied. However, you may change the Affiliation ID or select a different Affiliation Owner, if required.

Manage affiliation membership

About this task

On the Affiliation Membership screen, you 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 may add or remove connections from the affiliation.

Steps

- To add an SP partner connection to the affiliation, select the connection from the drop-down list and click **Add**.

> **Important:** Each connection in the affiliation must be configured to use the same IdP adapter instance for generating account links (see *Manage authentication source mappings* on page 513).

- To remove a member of the affiliation, click **Delete** under Action for the connection and click **Save**.

> **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.

Review an SP affiliation

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

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 via third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page.

Like other user-facing screens, 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 may 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, thus streamlining the registration process.

Depending on the requirements and configuration of existing components, the configuration process may involve changes to these configuration components: authentication policy contracts, local identity profiles, HTML Form Adapter instances, and IdP authentication policies.

Set up PingDirectory for customer identities

About this task

PingFederate stores customer identities in PingDirectory™ . Once you have installed PingDirectory, update the LDAP schema with a new object class and a couple attributes to store customer identities and their connections. (An LDIF file is provided.) To optimize performance, updates in indexes should also be applied to the directory. In addition, you must configure in PingFederate an LDAP data store connection to your PingDirectory and an LDAP Username Password Credential Validator instance for the HTML Form Adapter to validate user credentials. If you have already created these components, you may reuse them.

> **Note:** Skip this configuration if your use case does not involve registration or profile management (see *Enable third-party identity providers without registration* on page 580).

Steps

1. Update the LDAP schema.
   a. Sign on to the PingDirectory administrative console.
   b. Go to the **LDAP Schema**# **Schema Utilities** screen.
   c. Click **Import Schema Element**.
   d. 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.
   e. 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 data store connection to your PingDirectory on the **System**# **Data Stores** screen.

   If you have already created an LDAP data store connection to your PingDirectory, you may reuse it.
4. Create an instance of the LDAP Username Password Credential Validator on the **System**# **Password Credential Validators** screen to validate user credentials stored in PingDirectory.

   If you have already created such LDAP Username Password Credential Validator instance, you may 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 (see *Configure LDAP base DN and attributes* on page 565).

Manage local identities profiles

About this task

When associated with an HTML Form Adapter instance, a local identity profile provides users the option to authenticate via third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page. A typical customer identity and access management (CIAM) use case only requires one local identity profile. As needed, you may create multiple profiles to suit the needs of your organization. Local identity profiles are defined on the **Identity Provider**# **Identity Profiles** screen.

Steps

- To configure a new profile, 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**.

Define a local identity profile

About this task

On the **Profile Info** screen, enter a name for the local identity profile and choose an authentication policy contract, which is the set of attributes that are returned from an authentication policy that uses this profile. In addition, choose whether to enable self-service registration and profile management.

Steps

1. Enter a name in the **Local Identity Profile Name** field.
2. Select a contract from the **Authentication Policy Contract** list.

   If you have not yet defined the desired contract, click **Manage Policy Contracts**.
3. 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.

   (This check box is not selected by default.)
4. Select the **Enable Profile Management** check box if you want to allow users to manage their accounts.

   (This check box is not selected by default.)

Define authentication sources

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.

Attributes received from third-party identity providers can optionally be stored 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. Configure authentication sources.
   - To add a new authentication source, enter the desired value in the field and click **Add**.

     Tip: If you use the authentication source names Facebook, Google, LinkedIn, Twitter, FIDO, or Windows Hello, the HTML Form Adapter default templates render the associated icons on the registration and profile management pages.
   - To modify an existing authentication source, use the **Edit**, **Update**, and **Cancel** workflow.
   - To remove an existing authentication source, click **Delete** for the applicable authentication source.

     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.

- To change the display order of the authentication sources on the sign-on page and the profile management page, use the up and down arrows to reorder them.

Make a note of the values defined here. In a later step, you will be creating a rule for each authentication source in an IdP authentication policy. Each rule forms a policy path that initiates the authentication process.

2. Configure storage settings for attributes received from third-party identity providers.

Inapplicable (and not shown) if neither registration nor profile management is enabled on the **Profile Info** screen.

a. If attributes should be stored, select the **Store Attributes** check box.

(This check box is not selected by default.)

b. If attributes should be retained after users disconnect third-party identity providers from their accounts, select the **Keep Attributes After Users Disconnect** check box.

(This check box is not selected by default.)

c. If attributes should be 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.

(The **Update Attributes When Users Authenticate** check box is not selected by default, and the **Minimum Number of Days Between Updates** field has no default value.)

Results

Define local identity fields

About this task

On the **Fields** screen, define the local identity fields that suit your registration and profile management requirements. 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

- To add a new local identity fields, click **Create New Field**.
- To select one of the local identity fields as the unique identifier, select the **Unique ID** option for the applicable field.

  Applicable and required only if registration is enabled on the **Profile Info** screen.

Tip: Any field that uses the **Checkbox**, **Checkbox Group**, **Date**, or **Dropdown** input control cannot be chosen as the unique identifier because values from such field will likely collide as the population of users grows.

- To modify an existing local identity field, click **Edit** for the applicable field.
- To remove an existing local identity field or to cancel the removal request, click **Delete** or **Undelete** for the applicable field.
- To change the display order of the local identity fields on the registration page and the profile management page, use the up and down arrows to reorder them.
- To mask local identity field values in logs for the configuration scenario where OGNL expressions might be used to map derived values into outbound SSO tokens in authentication policies, select the **Mask all OGNL-expression generated log values** check box.

Configure a local identity field

About this task

On the **Field Configuration** screen, create a new or modify an existing local identity field.

Steps

1. Enter an identifier under **ID**.

   Note that you cannot change the identifier of an existing field.
2. Enter a name under **Label**.

   This is the field name that users see on the registration and profile management pages.
3. Select one of the following input controls from the list under **Type**.
   - **Checkbox**
   - **Checkbox Group**
   - **Date**
   - **Dropdown**
   - **Email**
   - **Phone**
   - **Text**
   - **Hidden**
4. Select whether this field should appear on the registration page, the profile management page, or both under **Applies To**.

   Applicable only if both registration and profile management are enabled on the **Profile Info** screen. Both pages are selected by default.

   If only registration (or profile management) is enabled, all fields, with the exception of hidden fields, are shown on the registration page (or the profile management page).
5. Optional: Select the relevant parameters under **Parameters**.

   You can make a non-hidden field mandatory or read-only. You can also configure PingFederate not to record values from this field in logs.
6. 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 (see *Configuring local identity mapping* on page 351).

   Not shown if you have chosen an input control of **Checkbox group**, **Email**, **Phone**, or **Hidden**, or the **Read-Only** parameter.
7. Add the applicable predefined value (or values) under **Options**.

   Applicable and required only if you have chosen **Checkbox Group** or **Dropdown** as the input control.
8. Click **Done**.

   The administrative console brings back the **Field** screen, where you can configure other options and save your changes.

Configure email ownership verification options

About this task

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 is optional and can be configured on a per-local identity profile basis.

When enabled, PingFederate generates a notification message for email ownership verification as the user submits the registration request. The email-verification message is valid for a configurable amount of time, 24 hours by default. If the user cannot find the previously sent message, the user can request another one by accessing the email ownership verification endpoint. Moreover, if profile management is enabled, the profile management page displays a reminder until the user verifies the associated email address as well. Like other local identity fields, the email verification status is stored in the directory and can be relayed to the applicable target applications through IdP authentication policies.

Steps

1. Select the **Enable Email Ownership Verification** check box if you want to offer users the opportunity to verify the email address associated with their accounts.

   This check box is not selected by default.

   > **Note:** The rest of the steps are applicable only if email ownership verification is enabled.

2. Select a field from the **Email Address Field** list.

   The field value represents the recipient of the verification message.

   Only fields that use the **Email** or **Text** input control are eligible and shown.
3. Select a field from the **Ownership Status Field** list.

   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. Once the user verifies the email ownership, PingFederate sets the value to `true`.

   Only fields that use the **Hidden** input control are eligible and shown.
4. If you want to modify the longevity of the link in the email-verification message, update the **One-Time Link Lifetime** field.

   The default value is `1440` in minutes (24 hours).
5. Optional: If you want to use different template files for various events, update the applicable template fields.

   These templates are only applicable when using an SMTP Notification Publisher instance to deliver email-verification messages.

   Default template files are documented in the following table.

Template field	Default value
Email Template	message-template-email-ownership-verification.html
Sent Template	local.identity.email.verification.sent.html
Success Template	local.identity.email.verification.success.html
Error Template	local.identity.email.verification.error.html

   Note that the email template file is located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory while the rest can be found in the `template` directory.
6. Select a notification publisher instance from the list.

   If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.

Configure registration options

About this task

Configure the registration experience and specify the template file for the registration page.

Steps

1. If you want to enable invisible reCAPTCHA from Google to prevent automated registration attempts, select the **CAPTCHA** check box, and then click **Manage CAPTCHA Settings**.

   (This check box is not selected by default.)
2. If you want to use a different template file, update the **Registration Template** field.

   (The default value is `local.identity.registration.html`.)

Configure profile management options

About this task

Configure the profile management experience and specify the template file for the profile management page.

Steps

1. 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.
2. If you want to use a different template file, update the **Profile Template** field.

   (The default value is `local.identity.profile.html`.)

Manage data store configuration

About this task

Configure the data store where local identities are stored.

Steps

- Click **Configure Data Store** to begin.

Select a data store for customer identities

About this task

PingFederate stores customer identities in PingDirectory™ .

Steps

On the **Data Store** screen, select the desired LDAP data store from the list.

If you have not yet created an LDAP data store to connect PingFederate to your PingDirectory or if you want to review your LDAP data store settings, click **Manage Data Store**.

Configure LDAP base DN and attributes

About this task

On the **LDAP Configuration** screen, 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 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, it is recommended that the base DN here matches 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. Specify a base DN.
2. Optional: Click **View Local Identity Fields** to determine which attributes from the directory server should be added to the local identity profile.
3. Select a root object class, select an applicable attribute, and then click **Add Attribute**.

   Repeat this step to add more attributes as needed.

Configure LDAP relative DN and object class

About this task

On the **Identity Creation** screen, configure the settings for creating local identities. Enter a relative DN (RDN) pattern using the available fields and attributes, and then select an object class from the list.

When a user submits a registration request, PingFederate formulates the DN of the user by prefixing the RDN to the base DN defined on the **LDAP configuration** screen, 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. Enter a valid RDN pattern.

The pattern is:

`attribute1=`*`value1`*`[, ..., 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; for example:

```
entryUUID=${entryUUID}
```

3. Select an object class from the list.

Define data store mapping configuration

About this task

Configure the mapping between the local identity profile fields and the data store attributes.

Steps

- Select an LDAP attribute under **Data Store Attribute** for each local identity field.

Review data store configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review a local identity profile

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure the HTML Form Adapter for customer identities

About this task

After defining a local identity profile, associate it with an instance of the HTML Form Adapter so that PingFederate can leverage the HTML Form Adapter to present users the options to authenticate via third-party identity providers, self-register as part of the sign-on experience, and manage their accounts through a self-service profile management page.

For registration and profile management, ensure the HTML Form Adapter instance is configured to validate credentials stored in PingDirectory™ . This validation configuration however is not required if your use case does not involve registration or profile management (see *Enable third-party identity providers without registration* on page 580).

Steps

1. Go to the **Identity Provider# Adapters** screen.
2. Create a new HTML Form Adapter instance or reuse an existing instance by clicking on its name.
3. On the **IdP Adapter** screen, add 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** screen, select a local identity profile from the **Local Identity Profile** list.
5. Complete the rest of the configuration and save all changes.

Set up self-service registration

About this task

PingFederate leverages the HTML Form Adapter to deliver a secure and easy-to-use customer authentication, registration, and profile management solution. A typical self-service registration setup involves five components:

- A PingDirectory™ installation (*step 1*)
- An authentication policy contract (*step 2*)
- A local identity profile (*step 3*)
- An HTML Form Adapter instance (*step 4*)
- An IdP authentication policy (*step 5*)

To illustrate the configuration steps, consider the following example:

You are tasked to support a consumer registration use case, where users can 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.

Configuration steps:

Steps

1. Install PingDirectory, update its LDAP schema, set up the required index, and then create an LDAP data store and an LDAP Username Password Credential Validator instance in PingFederate.

   (For more information, see .)
2. Create an authentication policy contract using the **Identity Provider# Policy Contract** configuration wizard. Extend the authentication policy contract with three additional attributes; for example, firstName, lastName and mobileNumber.

   (For more information, see .)

3. Create a local identity profile using the **Identity Provider**# **Identity Profiles** configuration wizard.

a. On the **Profile Info** screen, enter a name of the local identity profile, select the authentication policy (from *step 2*), and select the **Enable Registration** check box.

b. On the **Authentication Sources** screen, click **Next**.

c. On the **Fields** screen, define four local identity fields, as follows:

Type	ID	Label	Parameters
Email	lipEmail	Email address	Select the **Required** and **Unique ID Field** check boxes.
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.

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.)

d. On the **Email Verification** screen, click **Next**.

e. On the **Registration** screen, click **Next**.

f. On the **Data Store Configuration** screen, click **Configure Data Store**.

g. On the **Data Store Configuration**# **Data Store** screen, select the LDAP data store that has been set up to connect to your PingDirectory.

h. On the **Data Store Configuration**# **LDAP Configuration** screen, 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**.

i. On the **Data Store Configuration**# **Identity Creation** screen, define the RDN pattern in the **Relative DN Pattern** field and select your object class (aPerson for this sample use case) from the **Object Class** list.

The pattern is:

```
attribute1=value1[, ..., 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; for example:

```
entryUUID=${entryUUID}
```

j. On the **Data Store Configuration**# **Data Store Mapping** screen, configure the mapping between the local identity profile fields and the data store attributes as follows:

Field	Data Store Attribute
lipEmail	mail
lipFirstName	givenName
lipLastName	sn
lipMobile	mobile

k. On the **Data Store Configuration**# **Summary** screen, click **Done**.

l. On the **Summary** screen of the local identity profile, click **Save**.

(For more information, see *Define a local identity profile* on page 560.)

4. Configure an HTML Form Adapter instance for customer identities.
   a. Go to the **Identity Provider# Adapters** screen.
   b. Create a new HTML Form Adapter instance or reuse an existing instance by clicking on its name.
   c. On the **IdP Adapter** screen, add the LDAP Username Password Credential Validator instance that has been set up to validate credentials stored on your PingDirectory.
   d. On the **IdP Adapter** screen, select the newly created local identity profile from the **Local Identity Profile** list.
   e. Complete the rest of the configuration and save all changes.

   (For more information, see .)
5. Create an IdP authentication policy.
   a. Go to the **IdP Providers# Policies** screen.
   b. Select the HTML Form Adapter instance (configured in *step 4*) under **Action**.
      1. For its **Fail** path, select **Done**.
      2. For its **Success** path, select the local identity profile (created in *step 3*).
   c. Click **Local Identity Mapping** underneath the selected local identity profile, which opens the **Inbound Mapping & Contract Fulfillment** configuration wizard.
   d. On the **Inbound Mapping & Contract Fulfillment# Inbound Mapping** screen, 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** screen as follows:

Inbound Mapping Fulfillment	Source	Value
pf.local.identity.unique.id	Adapter	username

   e. On the **Inbound Mapping & Contract Fulfillment# Attribute Sources & User Lookup** screen, click **Next**.
   f. On the **Inbound Mapping & Contract Fulfillment# Attribute Sources & User Lookup** screen, click **Next**.
   g. On the **Inbound Mapping & Contract Fulfillment# Issuance Criteria** screen, click **Next**.
   h. On the **Inbound Mapping & Contract Fulfillment# Summary** screen, click **Done**.

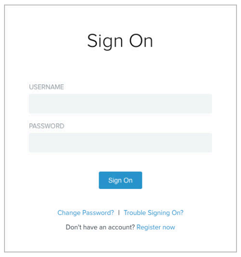
      The **Inbound Mapping & Contract Fulfillment** configuration wizard brings back the **Manage Authentication Policies** screen.
   i. Select the **IdP Authentication Policies** check box.

      Note: Other IdP authentication policies (if any) are enabled as well.
   j. Click **Save** to keep your changes.

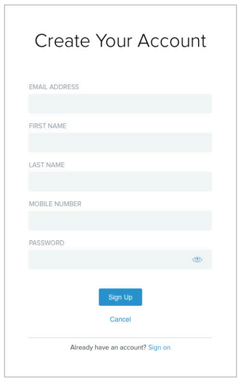
   (For more information, see *Applying policy contracts or identity profiles to authentication policies* on page 349.)
6. Map the authentication policy contract to the applicable Browser SSO connections, OAuth grant-mapping configuration, or both (see *Manage authentication source mappings* on page 513 and *Manage authentication policy contract grant mapping* on page 436).

Results

You have now successfully set up self-service registration. When users sign on through this HTML Form Adapter instance, they have the option to 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, the following registration page is presented:



Enable third-party identity providers

About this task

For registration, you can optionally 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, thus streamlining the registration process. 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.

- IdP connections or IdP adapter instances
- A PingDirectory™ installation (*step 1*)
- An authentication policy contract (*step 2*)
- A local identity profile (*step 3*)
- An HTML Form Adapter instance (*step 4*)
- An IdP authentication policy (*step 5*)

To illustrate the configuration steps, consider the following example:

You are tasked to support a consumer registration use case, where users can 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.

Configuration steps:

> ⓘ **Tip:** If you are familiar with the steps to set up PingDirectory to connect with PingFederate and an authentication policy contract (as documented in *Set up self-service registration* on page 567), you may skip to *step 3* to create a local identity profile.

Steps

1. Install PingDirectory, update its LDAP schema, set up the required index, and then create an LDAP data store and an LDAP Username Password Credential Validator instance in PingFederate.

   (For more information, see .)
2. Create an authentication policy contract using the **Identity Provider**# **Policy Contract** configuration wizard. Extend the authentication policy contract with three additional attributes; for example, firstName, lastName and mobileNumber.

   (For more information, see .)
3. Create a local identity profile using the **Identity Provider**# **Identity Profiles** configuration wizard.
   a. On the **Profile Info** screen, enter a name of the local identity profile, select the authentication policy (from *step 2*), and select the **Enable Registration** check box.
   b. On the **Authentication Sources** screen, enter `ACME` under **Authentication Source** and then 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 screen.

   > ⓘ **Tip:** If you are familiar with the steps to set up a local identity profile and an HTML Form Adapter instance for customer identities (as documented in *Set up self-service registration* on page 567), you may skip to *step 5* to create an IdP authentication policy.

   c. On the **Fields** screen, define four local identity fields, as follows:

Type	ID	Label	Parameters
Email	lipEmail	Email address	Select the **Required** and **Unique ID Field** check boxes.
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 the **Mask all OGNL-expression generated log values** check box (for all fields).
   d. On the **Email Verification** screen, click **Next**.
   e. On the **Registration** screen, click **Next**.
   f. On the **Data Store Configuration** screen, click **Configure Data Store**.
   g. On the **Data Store Configuration**# **Data Store** screen, select the LDAP data store that has been set up to connect to your PingDirectory.
   h. On the **Data Store Configuration**# **LDAP Configuration** screen, 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**.

i. On the **Data Store Configuration# Identity Creation** screen, define the RDN pattern in the **Relative DN Pattern** field and select your object class (aPerson for this sample use case) from the **Object Class** list.

   The pattern is:

   ```
   attribute1=value1[, ..., 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; for example:

   ```
   entryUUID=${entryUUID}
   ```

j. On the **Data Store Configuration# Data Store Mapping** screen, configure the mapping between the local identity profile fields and the data store attributes as follows:

Field	Data Store Attribute
lipEmail	mail
lipFirstName	givenName
lipLastName	sn
lipMobile	mobile

k. On the **Data Store Configuration# Summary** screen, click **Done**.

l. On the **Summary** screen of the local identity profile, click **Save**.

(For more information, see *Define a local identity profile* on page 560.)

4. Configure an HTML Form Adapter instance for customer identities.

a. Go to the **Identity Provider# Adapters** screen.

b. Create a new HTML Form Adapter instance or reuse an existing instance by clicking on its name.

c. On the **IdP Adapter** screen, add the LDAP Username Password Credential Validator instance that has been set up to validate credentials stored on your PingDirectory.

d. On the **IdP Adapter** screen, select the newly created local identity profile from the **Local Identity Profile** list.

e. Complete the rest of the configuration and save all changes.

(For more information, see .)

5. Create an IdP authentication policy.

a. Go to the **IdP Providers# Policies** screen.

b. Select the HTML Form Adapter instance (configured in *step 4*) under **Action**.

   1. For its **Fail** path, select **Done**.

   2. For its **Success** path, select the local identity profile (created in *step 3*).

c. 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 are familiar with the steps to setup the inbound mapping and contract fulfillment of an authentication policy contract through a local identity profile (as documented in *Set up self-service*

> *registration* on page 567), you may skip to *step i* to create a rule for the scenario where users choose to register and subsequently authenticate via ACME.

d. On the **Inbound Mapping & Contract Fulfillment**# **Inbound Mapping** screen, 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** screen as follows:

Inbound Mapping Fulfillment	Source	Value
pf.local.identity.unique.id	Adapter	username

e. On the **Inbound Mapping & Contract Fulfillment# Attribute Sources & User Lookup** screen, click **Next**.
f. On the **Inbound Mapping & Contract Fulfillment# Attribute Sources & User Lookup** screen, click **Next**.
g. On the **Inbound Mapping & Contract Fulfillment# Issuance Criteria** screen, click **Next**.
h. On the **Inbound Mapping & Contract Fulfillment# Summary** screen, click **Done**.

The **Inbound Mapping & Contract Fulfillment** configuration wizard brings back the **Manage Authentication Policies** screen.

Note: The remaining steps configure the fulfillment of the authentication policy contract for the scenario where users choose to register and subsequently authenticate via ACME.

i. Click **Rules** underneath the **Success** path of the HTML Form Adapter instance.
j. 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 follows:

Attribute Name	Condition	Value	Result
policy.action	equal to	ACME **Important:** The value here must match the value defined on the **Authentication Sources** screen (see *step 3b*).	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** screen.

Important: If you have defined multiple third-party identity providers on the **Authentication Sources** screen, 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 **Manage Authentication Policies** screen.

k. 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 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 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 in maintenance releases 9.2.3 and 9.3.3.

2. For its **Success** path, select the local identity profile (created in *step 3*).

l. Click **Local Identity Mapping** underneath the selected IdP connection, which opens the **Inbound Mapping & Contract Fulfillment** configuration wizard.

m. On the **Inbound Mapping & Contract Fulfillment**# **Inbound Mapping** screen, 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, namely the subject identifier from the IdP connection.

For this sample use case, the **Inbound Mapping** screen is configured as follows:

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

n. On the **Inbound Mapping & Contract Fulfillment**# **Attribute Sources & User Lookup** screen, click **Next**.

o. On the **Inbound Mapping & Contract Fulfillment**# **Attribute Sources & User Lookup** screen, click **Next**.

p. On the **Inbound Mapping & Contract Fulfillment**# **Issuance Criteria** screen, click **Next**.

q. On the **Inbound Mapping & Contract Fulfillment**# **Summary** screen, click **Done**.

The **Inbound Mapping & Contract Fulfillment** configuration wizard brings back the **Manage Authentication Policies** screen.

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.

r. Select the **IdP Authentication Policies** check box.

Note: Other IdP authentication policies (if any) are enabled as well.

s. Click **Save** to keep your changes.

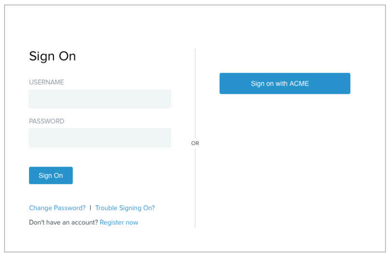
(For more information, see *Applying policy contracts or identity profiles to authentication policies* on page 349.)

Results

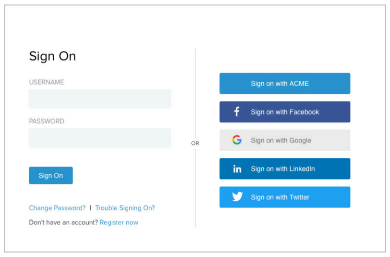
You have now successfully set up self-service registration with an option for users to register and subsequently authenticate via ACME. When users sign on through this HTML Form Adapter instance, they have two registration options:

- 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.

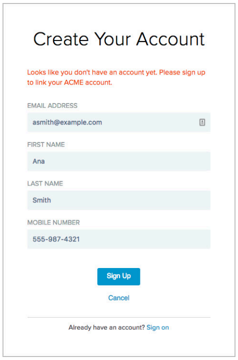
Based on the configuration of this sample use case, the following sign-on page is presented:


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.

Enable profile management

About this task

Besides registration, you may enable self-service profile management and specify which local identity fields users can update on the profile management page.

To illustrate the configuration steps, consider the sample use case in *Set up self-service registration* on page 567 or *Enable third-party identity providers* on page 570 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 *Set up self-service registration* on page 567 and *Enable third-party identity providers* on page 570 as well. If you require more information for a given step, refer to the same step in one of the aforementioned pages.

Steps

1. Set up PingDirectory™ to connect with PingFederate.
2. Create an authentication policy contract.
3. Configure profile management when creating a new or reconfiguring an existing local identity profile using the **Identity Provider# Identity Profiles** configuration wizard.
   a. On the **Profile Info** screen, select the **Enable Profile Management** check box.
   b. Optional: On the **Authentication Sources** screen, define authentication sources.
   c. On the **Fields** screen, select the **Profile Management** check box under **Show on** 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 the **Profile Management** check box for the lipMobile local identity field.
   d. On the **Email Verification** screen, click **Next**.
   e. On the **Registration** screen, click **Next**.
   f. On the **Profile Management** screen, select the **Enable Profile Deletion** check box.

      Generally speaking, this is an optional feature. It is selected here because it is one of the requirements of this sample use case.
   g. Continue from step 3f as documented in both *Set up self-service registration* on page 567 or *Enable third-party identity providers* on page 570.
4. Configure an HTML Form Adapter instance for customer identities.
5. Create an IdP authentication policy.
6. Provide the profile management URL to users.
   a. Go to the **Identity Provider# Identity Profiles** screen.
   b. Select the local identity profile that you have configured profile management in *step 3*.
   c. Copy the profile management URL as shown on the **Summary** screen and pass it to the users.

Results

You have now successfully enabled profile management. Authenticated users can review and modify the local identity fields that have been configured to be shown on the profile management page and delete their local accounts if the option to do so has been enabled.

The following screen capture provides a sample of the profile management page based on the sample use case:

Suppose you have added 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:

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.

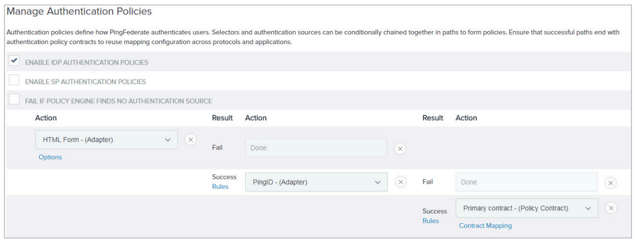
Create advanced registration mapping

About this task

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: username and policy.action. At runtime, regardless of whether 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.

To illustrate the configuration steps, consider the following setup that you have already made:

- A PingDirectory™ installation with a set of users.
- An LDAP data store, 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 multifactor authentication in multiple browser-based SSO use cases via SP connections, OAuth authorization code flow, and OAuth implicit flow. The following screen 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

You are now tasked to add support for a consumer registration use case similar to the one in *Set up self-service registration* on page 567, and at the same time keep the policy that enforces the multifactor 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 using the **Identity Provider# Identity Profiles** configuration wizard.
   a. On the **Profile Info** screen, enter a name of the local identity profile, select the authentication policy (from *step 2*), and select the **Enable Registration** check box.

      If you want to enable profile management as well, select the relevant check box.
   b. Complete the rest of the configuration to create the local identity profile.
4. Configure the HTML Form Adapter instance for customer identities.
   a. On the **IdP Adapter** screen, select a local identity profile from the **Local Identity Profile** list.
   b. Complete the rest of the configuration and save all changes.

5. Modify your existing IdP authentication policy.

a. Click **Rules** underneath the **Success** path of the HTML Form Adapter instance.

b. On the **Rules** dialog, create a policy path for users who choose to register. For this sample use case, configure as follows:

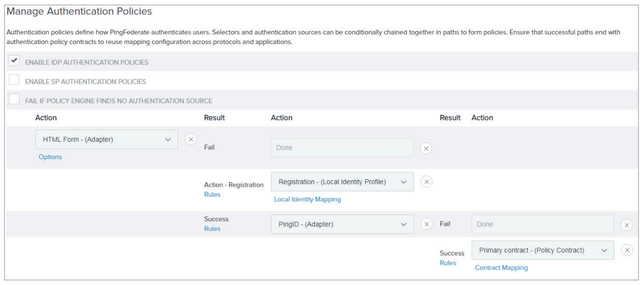
Attribute Name	Condition	Value	Result
policy.action	equal to	identity.registration	Registration The **Result** field controls the label shown for the policy path of this rule.

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 multifactor authentication requirement after authenticating successfully against the HTML Form Adapter.

When finished, click **Done**, which brings you back to the **Manage Authentication Policies** screen.

c. 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.

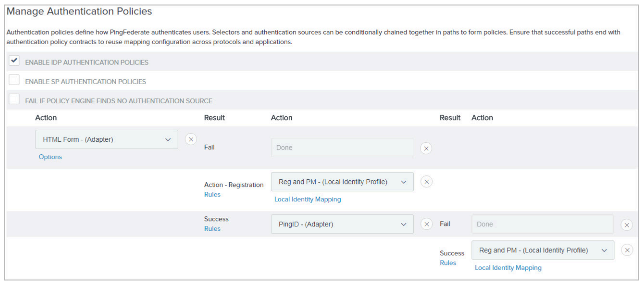
- [✓] ENABLE IDP AUTHENTICATION POLICIES
- [ ] ENABLE SP AUTHENTICATION POLICIES
- [ ] FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Action	Result	Action	Result	Action
HTML Form - (Adapter)	Fail	Done		
	Action - Registration	Registration - (Local Identity Profile)		
	Success	PingID - (Adapter)	Fail	Done
			Success	Primary contract - (Policy Contract)

d. 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

Action	Result	Action	Result	Action
HTML Form - (Adapter)	Fail	Done		
	Action - Registration	Reg and PM - (Local Identity Profile)		
	Success	PingID - (Adapter)	Fail	Done
			Success	Reg and PM - (Local Identity Profile)

Note: No reconfiguration is required in your Browser SSO connections and OAuth grant-mapping configuration for your new policy to take effect.

e. Click **Save** to keep your changes.

Results

You have now successfully added the requested consumer registration (and profile management) use case to your current policy.

Enable third-party identity providers without registration

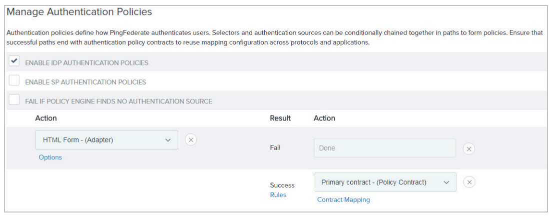
About this task

If you have already configured 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 via these providers. This setup involves the following components.

- IdP connections or IdP adapter instances configured to connect with your third-party identity providers
- 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 setup that you have already made.

- An HTML Form Adapter instance to validate local user credentials.
- An authentication policy contract.
- An IdP authentication policy that chains the HTML Form Adapter instance and an authentication policy contract so that the policy contract can harness attribute values returned by the HTML Form Adapter instance for multiple browser-based SSO use cases via SP connections, OAuth authorization code flow, and OAuth implicit flow. The following screen 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
HTML Form - (Adapter)	Fail	Done
	Success	Primary contract - (Policy Contract)

You are now tasked to enhance the sign-on experience by giving users the option to authenticate using their existing accounts at ACME (a major social network). It happens that you have already established an IdP connection to this social network.

Configuration steps:

Steps

1. Verify the IdP connection returns the attributes required to complete the browser-based SSO use cases.

   > **Tip:** As needed, you may also deploy and configure Cloud Identity Connectors to support identities from Facebook, Google, LinkedIn, or Twitter.

2. Make a note of which authentication policy contract is currently being used in your policy.

3. Create a local identity profile using the **Identity Provider#** **Identity Profiles** configuration wizard.

a. On the **Profile Info** screen, enter a name of the local identity profile and select the authentication policy (from *step 2*).

b. On the **Authentication Sources** screen, enter `ACME` under **Authentication Source** and then 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 screen.

4. Configure the HTML Form Adapter instance for customer identities.

a. On the **IdP Adapter** screen, select a local identity profile from the **Local Identity Profile** list.

b. Complete the rest of the configuration and save all changes.

5. Modify your existing IdP authentication policy.

a. Click **Rules** underneath the **Success** path of the HTML Form Adapter instance.

b. On the **Rules** dialog, create a policy path for users who choose to authenticate via ACME. For this sample use case, configure as follows:

Attribute Name	Condition	Value	Result
policy.action	equal to	ACME ⓘ **Important:** The value here must match the value defined on the **Authentication Sources** screen (see *step 3b*).	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** screen.

> ⓘ **Important:** If you have defined multiple third-party identity providers on the **Authentication Sources** screen, 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 for this sample use case where users can also authenticate using their local accounts.

When finished, click **Done**, which brings you back to the **Manage Authentication Policies** screen.

c. 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

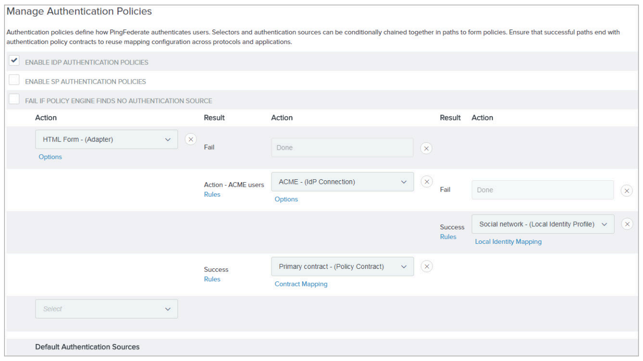
opportunity to do over. When triggered, the policy engine routes the requests back to the first checkpoint of the invoked authentication policy.

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.

2. For its **Success** path, select the local identity profile (created in *step 3*) and then completes its **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).

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

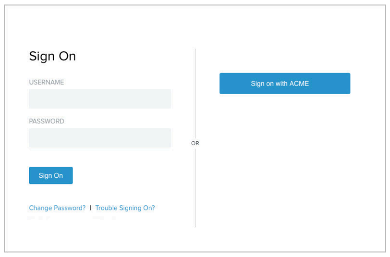
Action	Result	Action	Result	Action
HTML Form - (Adapter) Options	Fail	Done		
	Action - ACME users Rules	ACME - (IdP Connection) Options	Fail	Done
			Success Rules	Social network - (Local Identity Profile) Local Identity Mapping
	Success Rules	Primary contract - (Policy Contract) Contract Mapping		
Select				

Default Authentication Sources

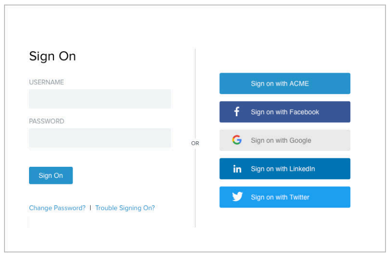
d. Click **Save** to keep your changes.

Results

You have now successfully added the option to authentication via ACME without enabling registration. When users sign on through this HTML Form Adapter instance, the following sign-on page is presented.



If you have added Facebook, Google, LinkedIn, and Twitter as the authentication sources, the following sign-on page is presented.



Users can sign on using their local accounts or third-party identity provider accounts.

Service provider SSO configuration

In an SP role, you use the PingFederate administrative console to configure local application-integration information and to manage connections to your IdP-partner sites. Prior to configuring connections to IdPs, you must establish your site as an SP on the **System**# **Protocol Settings**# **Roles & Protocols** screen.

Note that only one connection is needed per partner, even if you are integrating more than one web application.

While your entity ID is defined on the **System**# **Protocol Settings**# **Federation Info** screen, you may identify your organization differently through the use of virtual server IDs on a per-connection basis (see *Multiple virtual server IDs* on page 133).

Additionally, you may deploy an SP connection to bridge a service provider to one or more identity providers through one or more authentication policy contracts (see *Federation hub use cases* on page 127 and *Federation hub and authentication policy contracts* on page 131 for more information).

Note: This topic applies to configuration settings needed for browser-based SSO. While there is some cross-over information also applicable to WS-Trust STS, if you are using PingFederate *exclusively* as an STS, start with *WS-Trust STS configuration* on page 673.

SP application integration settings

The integration of local applications with PingFederate is the essential "last-mile" configuration that allows end-users at your IdP partner's website to access your protected resources. This process is facilitated through the use of application-integration kits and a robust Software Development Kit (see *SSO integration kits and adapters* on page 112).

Under **Integration** on the **Service Provider** menu, you configure the SP adapters that PingFederate uses to create user sessions that allow SSO access to your protected resources. You can also set Default URLs to which users may be directed during SSO or SLO, and you can 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 STS, the selections under **Integration** also include menu items for configuring plug-in **Token Generators** (see *Service provider STS configuration* on page 691).

Manage SP adapters

About this task

An SP adapter is used to create a local-application session for a user in order for PingFederate to provide SSO access to your applications or other protected resources. You must configure at least one instance of an SP adapter in order to set up connections to IdP partners. You can also 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 also deploy additional integration kits from the Ping Identity Downloads website.

You manage SP adapter instances on the **Service Provider**# **Adapters** screen.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Results

Important: After installing new adapter program files, you may be required to make additional configuration changes in areas such as adapter instances and connections. as prompted by the administrative console.

Note: Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.

If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

Create an SP adapter instance

About this task

The first step in creating an adapter instance is choosing an adapter type.

Steps

- On the **Type** screen, configure the basics of this adapter instance.
  a. Enter the required information and select the adapter type from the list.
  b. Optional: 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 screens.

Configure an SP adapter instance

About this task

Depending on the selected adapter, the **IdP Adapter** screen presents you with different configuration parameters.

Steps

- Follow the on-screen instructions to configure the adapter instance.

  Note: 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, refer to *Configure an OpenToken SP Adapter instance* on page 738 for configuration information.

  If you are configuring an adapter from an integration kit (including any SaaS connector), locate the user guide from our *PingFederate documentation* website and configure the adapter instance accordingly.

Invoke SP adapter actions

About this task

Adapters can be written to 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

- Follow the on-screen instructions to complete the actions required.

Extend an SP adapter contract

About this task

Adapters may be written with an option allowing administrators to add to the attributes required for creating usable sessions. This feature might be needed, for example, 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 check box to modify the configuration.

Steps

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Identify the target application

About this task

On the **Target App Info** screen, enter the name of the target application and the URL of the application icon, accessible through the IdP Adapter interface (`IdpAuthenticationAdapterV2`) in the PingFederate Java SDK. (For more information about the SDK, see *SDK Developer's Guide* on page 931.) Both fields are optional.

Note: If this is a child instance, select the override check box to modify the configuration.

Steps

1. Optional: Enter the application name in the field.
2. Optional: Enter the URL to the application icon in the field.

Review an SP adapter configuration

About this task

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Configure target URL mapping

About this task

When you have more than one target session defined in an IdP connection, you must map the target URL to its target session. When PingFederate receives an SSO or 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 Redirect Validation settings. If you have not done this, follow the instructions in *Configure redirect validation* on page 316.

For example, this mapping configuration may 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. It is worth noting that when this information is provided with the SP request, the mapping table is ignored (see *SP services* on page 761).

Furthermore, 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 (see *Federation hub use cases* on page 127).

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 **Service Provider**# **Target URL Mapping** screen. The configuration process involves entering an URL and select a target session for it. Refer to the following table for more information.

Field	Description
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 may 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:** If a target session is not allowed based on restrictions imposed (see *Restrict a target session to certain virtual server IDs* on page 608), PingFederate tries the next entry.
Target Type	The type of the **Target Session**. 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**.
Target Session	A selection of configured SP adapter instances or SP connections. The available values depends on the chosen the **Target Type** list.

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.
2. Select a target type from the list.

   Applicable only when the IdP role is activated with at least one protocol for browser-based SSO.
3. Select a target session from the list.
4. Click **Add Mapping**.
5. Repeat these steps to add multiple mappings.

Results

Use the up and down arrows to re-arrange the order of the mappings. Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to a mapping. Use the **Delete** and **Undelete** workflow to remove a mapping or cancel the removal request.

Configure Identity Store Provisioners

PingFederate allows you to create custom Identity Store Provisioners to bridge the inbound 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 Software Developer Kit for PingFederate, you can create and test these custom Identity Store Provisioners (see the PingFederate *SDK Developer's Guide on page 931*).

To support custom attributes, you must add the schema extension and the custom attributes to the 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 *Define custom SCIM attributes* on page 638 and *SCIM 1.1 Core Schema* (www.simplecloud.info/specs/draft-scim-core-schema-01.html).

Note: The Identity Store Provisioner option is active only after you enable the **Inbound Provisioning** protocol on the **System# Protocol Settings# Roles & Protocols** screen (see *Choosing Roles and Protocols*).

Note: Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.

If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

Create an Identity Store Provisioner instance

About this task

On the **Type** screen, you begin creating an instance of an identity store that PingFederate uses to bridge the inbound SCIM processing to an external user store using a custom implementation.

Steps

1. Enter the a name and an ID.
2. Select a provisioner type from the list.

   Available provisioner types are limited to those that are currently installed on your server.
3. Optional: Select a **Parent Instance** from the 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.

Define the Identity Store Provisioner behavior

About this task

Different configuration parameters are available on the **Identity Store Provisioner** screen. These options are controlled by the provisioner plug-in software (see topics about the Identity Store Provisioner interfaces in the *SDK Developer's Guide* on page 931 for more information).

Extend the Identity Store Provisioner contract

About this task

Identity Store Provisioners may be written with an option allowing administrators to add to the core attributes the plug-in instance requires. Both the core and extended contract attributes you define here must be mapped when you configure “Write Users” within an inbound provisioning connection.

Tip: To keep your plug-in 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

- To add an attribute, enter the attribute name in the text box and click **Add**.
- To modify an attribute name, follow these steps:
  a. Click **Edit** under **Action** for the attribute.
  b. 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.

Extend the Identity Store Provisioner contract for groups

About this task

Identity Store Provisioners may be written with an option allowing administrators to add to the core group attributes the plug-in instance requires. Both the core and extended group attributes you define here must be mapped when you configure “Write Groups” within an inbound provisioning connection.

ⓘ **Tip:** To keep your plug-in 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

- To add an attribute, enter the attribute name in the text box and click **Add**.
- To modify an attribute name or masking selection, follow these steps:
  a. Click **Edit** under **Action** for the attribute.
  b. 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.

Review the Identity Store Provisioner configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Configure default URLs

About this task

As an SP, you can supply a default URL that the end-user may see when an SSO request succeeds (that is, a session is created at your site) but the target resource is not available or not specified.

ⓘ **Note:** You can also specify default target SSO URLs for individual IdP connections, which take precedence over this global setting (see *Configure default target URLs* on page 622).

Similarly, you can specify to prompt a default URL indicating a successful 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* on page 275. If localization is not needed, you may also specify a message directly in this field to change the default.

Your application or your partner's application may supply these URLs at runtime (see *SP services* on page 761), but if none is provided, PingFederate will use the default values you enter on this screen 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 (see *Customizable user-facing screens* on page 257).

View SP application endpoints

About this task

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate.

Steps

- Click **Application Endpoints** on the **Service Provider** menu 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 *SP services* on page 761 and *System-services endpoints* on page 773.

Federation settings

If your identity federation uses the SAML 2.0 XASP profile (see *Attribute Query and XASP*), you may need to identify the IdP connection to which an attribute request applies. If so, use the **Service Provider**# **Attribute Requester Mapping** screen to complete the configuration (see *Manage attribute requester mappings* on page 590).

You can view endpoints that your federation partners need to know to access your services under **Service Provider**# **Protocol Endpoints**.

Manage attribute requester mappings

About this task

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 (see *Attribute Query and XASP* on page 45). Optionally, an application may also supply an issuer DN, which can be used to determine the correct IdP (*Attribute Authority*) to use for a set of users associated with an IdP.

Note: The Format query parameter must be set to a specified value for XASP (see *SP services* on page 761).

You can map X.509 identifying information to connections and specify a default connection on the **Service Provider**# **Attribute Requester Mapping** screen. (Note that this screen is only presented if the XASP profile is enabled on the **System**# **Protocol Settings**# **Roles & Protocols** screen.)

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 may use a regular expression to match different DNs to the same connection. Only one expression may 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.
   a. Enter an issuer DN under **Issuer DN Pattern**.
   b. Select an IdP connection under **IdP Connection Name**.
   c. Click **Add**.
   d. Repeat these steps to add more entries.
2. Map one or more subject DNs to SAML 2.0 IdP connections, as needed.
   a. Enter an subject DN under **Subject DN Pattern**.
   b. Select an IdP connection under **IdP Connection Name**.
   c. Click **Add**.
   d. Repeat these steps to add more entries.
3. Select a default IdP connection from the list.

Results

Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an entry. Use the **Delete** and **Undelete** workflow to remove an entry or cancel the removal request.

View SP protocol endpoints

Click **Protocol Endpoints** in the **Service Provider** menu 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**# **Protocol Settings**# **Federation Info** screen. These endpoints are built into PingFederate and cannot be changed.

Your federation partners or STS clients need to know the applicable SP services endpoints to communicate with your PingFederate server. Configured service endpoints for SAML connections are included in metadata export files.

PingFederate provides a favorite icon for all protocol endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 291.

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** screen. 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 (see). For more information about Third Party Initiated Login flow, see the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ThirdPartyInitiatedLogin).
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.
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 **Important** 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.

Service	**URL and Description**

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 (see).
	Note: If multiple token-generator instances of the same type are configured for the same connection or token-to-token mapping, a query parameter, TokenGeneratorId, must be added to either of these endpoints—see .
	(See **Important** 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` (see).

Virtual server ID support

For SAML connections using multiple virtual server IDs (see *Multiple virtual server IDs* on page 133), each virtual server ID has its own set of protocol endpoints. You may export a connection metadata for your partner on the **System**# **Metadata Export** screen (see *Exporting connection-specific SAML metadata* on page 155).

For WS-Federation (and SAML) connections using multiple virtual server IDs, you may provide your partner the federation metadata endpoint (`/pf/federation_metadata.ping`) with the PartnerIdpId *and* vsid parameters; for 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
	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 may 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* on page 773.)

Note that 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 may pass the token-to-token endpoint to your partners as-is.

Manage IdP connections

As an SP, you manage connection settings to support the exchange of federation-protocol messages (OpenID Connect, SAML, WS-Federation, or WS-Trust) with an IdP, OAuth client, OpenID Provider (OP), or STS client application at your site.

These settings include:

- User attributes that you expect to receive in an SSO token (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 (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 (see).

As an SP, you respond to user requests for SSO and 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 (see *Manage target session mappings* on page 605).

(No adapter instance or authentication policy contract is required for a connection that uses only the Attribute Query profile. For more information, see *Manage Attribute Query profile* on page 625.)

If you intend to pass attribute values to an adapter instance from a local data store, you must define the data store during this configuration, if you have not done so already (see *Managing data stores* on page 166).

Administrative interface

You manage connection settings using the **IdP Connection** wizard, 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 screen, where you manage one or more settings. You may move to a sibling task using the **Next** or **Previous** button. If you are on a sub task, you may also move to its parent task using the **Done** button.

When creating a new connection, you may save your progress using the **Save Draft** button. Note that not all screens offer this option. When you reach the **Activation & Summary** screen, you must click **Save** to complete the new connection.

When editing an existing connection, you may make changes and then click **Save** to commit your changes. In order words, you are not required to step through all screen to reach the **Activation & Summary** screen before you can save your changes.

Note: The **Save** button is available on most screen. If a screen does not show a **Save** button, click **Next** or **Done** until you reach to a screen where you can use its **Save** button to commit your changes.

Access IdP connections

About this task

The **Service Provider** menu displays a list of the most-recently modified IdP connections. You may create or import a connection. You may also edit a recently modified connection by clicking on its connection name.

To access the rest of the connections, click **Manage All** to open the **IdP Connections** screen. The **IdP Connections** screen displays 20 connections at a time. As needed, you can 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, and default virtual server ID; narrow by protocol and status; and perform various connection-related tasks.

Steps

- To edit a connection, select the connection by its name. For the setting you want to make a change, select the corresponding screen title and then follow the configuration wizard to complete the task.
- To create a connection, follow the **Connection** configuration wizard to create a new connection to your IdP partner.
- To copy a connection, select **Copy** under **Action** and then follow the **Connection** 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.
- To export a connection, select **Export Connection** under **Action** 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 making changes to it.
- To import a connection, click **Import Connection** and 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 may modify the XML file to suit your needs. The XML file may also be imported to another PingFederate environment acting in the same federation role (SP) at your site. Note that the source and the target must run the same version of PingFederate.
- To export metadata for any SAML Browser SSO connection, click **Export Metadata** under **Action** and then follow the on-screen instructions to complete the task.
- To update a SAML Browser SSO connection, select **Update with Metadata** under **Action** and then follow the on-screen instructions to complete the task.

  You may update a connection via a metadata XML file or a metadata URL.

  Important: The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.
- To toggle the status of a connection, slide the toggle switch to enable or disable a connection.
- To remove a connection, select **Delete** under **Action**.

- To override the verbosity of runtime transaction logging for all IdP 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 IdP 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** 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 the **Identity Provider**# **SP Connections** screen as well.

  For more information about this advanced setting and its impact, see *Configuring automatic connection validation* on page 250.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Resolve IdP connection errors

About this task

PingFederate automatically validates configured connections before displaying them on the **Connections** screen. 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-screen instructions to modify the configuration, one connection at a time.

Choose an IdP connection type

About this task

Indicate on the **Connection Type** screen whether the connection to this partner is for Browser SSO, WS-Trust 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.

If you have selected multiple protocols on the **System**# **Protocol Settings**# **Roles & Protocols** screen, you must select the applicable protocol on the **Connection Type** screen 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. Note that each connection must use a unique (partner) connection ID.

Steps

- To configure a connection for secure browser-based SSO, select the **Browser SSO Profiles** check box and a protocol from the list (if necessary).
- To configure an STS connection, select the **WS-Trust STS** check box and the default token type from the list.
- To configure a connection that exchanges SAML assertions or JWTs for access tokens, select the **OAuth Assertion Grant** check box.

  Note that the **OAuth Assertion Grant** option is available only if at least one Access Token Manager instance has been configured on the **OAuth Server**# **Access Token Management** screen.

  For more information about these standards, see *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants* (tools.ietf.org/html/rfc7522) and *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants* (tools.ietf.org/html/rfc7523).
- To configure an inbound provisioning connection, make that selection 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) are preserved.

  Note that the **Inbound Provisioning** option is active only if the **Inbound Provisioning** protocol is enabled on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
- Optional: If your PingFederate license manages connections by groups, then you can select a group for this connection.

  This option is not displayed for unrestricted or other types of licenses.

Choose IdP connection options

About this task

On the **Connection Options** screen (shown only for browser-based SSO connections), you can enable browser-based SSO in conjunction with JIT provisioning. Additionally, you may also choose to map user attributes for persistent grants used by the optional PingFederate OAuth authorization server.

For SAML 2.0, you also have the option of configuring the **Attribute Query** profile, with or without the browser-based SSO.

Steps

- To create a connection for browser-based SSO, select the **Browser SSO** check box.
- To enable JIT provisioning, OAuth attribute mapping, or both for this connection, make that selections (after selecting the **Browser SSO** check box).

  Note that the **OAuth Attribute Mapping** option is only available when the OAuth 2.0 authorization server (AS) role is enabled on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
- To create a connection to facilitate the SAML 2.0 Attribute Query profile, select the **Attribute Query** check box (see *Attribute Query and XASP* on page 45).

Import 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.

The reload frequency is configurable through the **Reload Delay** field on the **System**# **Metadata Settings**# **Metadata Lifetime** screen. The default reload frequency is daily.

Although optional, it is recommended that you turn on notifications for SAML metadata update events on the **System**# **Runtime Notifications** screen.

> **Note:** If the metadata contains changes that require additional configuration, the notification message also provides a list of the applicable items.

After the connection is created, you can add, remove, or change the metadata URL associated with the connection in the **Import Metadata** screen. In addition, you can also toggle the **Enable Automatic Reloading** option 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 (see www.incommon.org/participants).

Steps

Refer to the following steps to import or update metadata. Instructions vary depending on the medium of the metadata.

Metadata medium	Steps
A metadata file	**a.** On the **Import Metadata** screen, select the **File** option. **b.** 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** screen, and then click **Next**. **c.** On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **d.** Click **Next**.

Metadata medium	Steps
A metadata URL	**a.** On the **Import Metadata** screen, select the **URL** option. **b.** 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**. **c.** Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. **d.** 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. **e.** Click **Next**.

Identify the partner

About this task

On the **General Info** screen, 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 screen you can define a default error message that end users see in the event that SSO fails.

Steps

1. Provide the basic information to identify your partner.

   Refer to 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 via 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 an STS-only connection, this ID can be any unique identifier.

Field	Description
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** screen.)
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 **System**# **Protocol Settings**# **Federation Info** screen, 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, please refer to 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.
Error Message (Applicable only to SAML or OpenID Connect connections)	If an error occurs on this server, the end user's browser may 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 may 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 can be 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.

In addition, the protocol settings can be refreshed 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 could be mapped to the target applications later. In the rare 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 may 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**.

   (Note that this step is not applicable to an STS-only connection.)

Populate extended property values

Steps

Optional: On the **Extended Properties** screen, add, remove, or update one or more values for any extended properties defined on the **System**# **Extended Properties** screen.

Applicable and shown only if one or more extended properties are defined on the **System**# **Extended Properties** screen.

Extended property values can serve as metadata. They can also help drive authentication requirements. To learn more, see .

Define additional issuers

About this task

On the **Additional Issuers** screen, define additional issuers that PingFederate can accept when validating ID tokens obtained through this connection. This screen appears only when the **Enable Additional Issuers** check box on the **General Info** screen is selected.

Steps

1. 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.

2. Optional: Define additional issuers.

   Applicable only when the **Accept All Issuers (Not Recommended)** is not selected.

   a. Enter the issuer under **Additional Issuer**,
   b. Optional: Enter information about the issuer under **Description**.

   The **Primary Issuer** represents the issuer defined on the **General Info** screen and is always accepted.

Configure SP Browser SSO

Browser-based SSO (also known as Browser SSO) relies on a user's web browser and HTTP requests to broker identity-federation messaging (in XML or JWT) between an IdP and an SP (in contrast to WS-Trust STS messaging, which is typically application-driven across the back channel and does not require browser mediation).

To continue, click **Configure Browser SSO**.

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 (see *Federation roles* on page 30). 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

- *Select SAML profiles* on page 603
- *Configure user-session creation* on page 603
  - *Choose an identity mapping method* on page 604
  - *Define an attribute contract* on page 604
  - *Manage target session mappings* on page 605
- *Manage protocol settings* on page 613
  - *Specify SSO service URLs (SAML)* on page 614
  - *Define SLO service URLs (SAML 2.0)* on page 616
  - *Select allowable SAML bindings (SAML)* on page 617
  - *Specify an artifact lifetime (SAML 2.0)* on page 617
  - *Define artifact resolver locations (SAML)* on page 618
  - *Configure default target URLs* on page 622
  - *Override authentication context in an IdP connection* on page 622
  - *Configure signature policy* on page 624
  - *Specify XML encryption policy (for SAML 2.0)* on page 624

SAML 1.x configuration steps

- *Select SAML profiles* on page 603
- *Configure user-session creation* on page 603
  - *Choose an identity mapping method* on page 604
  - *Define an attribute contract* on page 604
  - *Manage target session mappings* on page 605
- *Manage protocol settings* on page 613
  - *Specify SSO service URLs (SAML)* on page 614
  - *Select allowable SAML bindings (SAML)* on page 617
  - *Define artifact resolver locations (SAML)* on page 618
  - *Configure default target URLs* on page 622
  - *Configure signature policy* on page 624

WS-Federation configuration steps

- *Configure user-session creation* on page 603
  - *Choose an identity mapping method* on page 604
  - *Define an attribute contract* on page 604
  - *Manage target session mappings* on page 605
- *Manage protocol settings* on page 613
  - *Specify a service URL (WS-Federation)* on page 615
  - *Configure default target URLs* on page 622
  - *Configure signature policy* on page 624

OpenID Connect configuration steps

- *Configure user-session creation* on page 603
    - *Choose an identity mapping method* on page 604
    - *Define an attribute contract* on page 604
    - *Manage target session mappings* on page 605
- *Manage protocol settings* on page 613
    - *Configure OpenID Provider information* on page 619
    - *Configure default target URLs* on page 622
    - *Override authentication context in an IdP connection* on page 622

After configuring SSO settings, you will normally need to configure authentication credentials, the range of which depends on your SSO selections (see *Configure security credentials* on page 654). Also, other configuration tasks may remain to be configured for new or modified connections, depending on the selected options on the **Connection Options** screen.

Select SAML profiles

About this task

A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use.

For SAML 2.0, PingFederate supports all IdP- and SP-initiated SSO and SLO profiles. For SAML 1.x, PingFederate supports both the standard IdP-initiated SSO profile and a proprietary ("destination-first") SP-initiated SSO profile.

(For information on typical SAML SSO and SAML 2.0 SLO profile configurations, including illustrations, see *SAML 1.x profiles* on page 33 and *SAML 2.0 profiles* on page 36.)

Note that the **SAML Profiles** screen does not apply to OpenID Connect and WS-Federation IdP connections.

Steps

- Select the applicable profile (or profiles) based on your partner agreement.

  For SAML 2.0, you must always 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 screen is **SP-initiated SSO**.

Configure user-session creation

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 may associate one or more authentication policy contracts to the IdP connection (see for more information).

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

- To continue, click **Configure User-Session Creation**.

Choose an identity mapping method

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 (see). On the **Identity Mapping** screen, you choose which method to use in this IdP connection. You and your partner should decide in advance which option to use (see).

If your site is using account linking, then establishing an attribute contract is not required. Depending on your partner agreement, however, you may 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 (see).

> **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

- If you want to dynamically associate remote users with local accounts using a known attribute to identify a user (for example, 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** (see).
- If you want to create a long-term association between a remote user and a local account, select **Account Linking**.

  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.

  > **Tip:** PingFederate uses a default, HSQLDB database to handle account linking. You can use your own database instead, as needed. For more information, see .
- 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**. Additionally, select **No Mapping** if you are deploying an IdP connection solely for OAuth attribute mapping without the use of an authentication policy contract (see *Configure IdP connection grant mapping* on page 431).

Define an attribute contract

About this task

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in SSO tokens for this connection (see *Attribute contracts* on page 121). You may extend the attribute contract with additional attributes. Optionally, you can configure PingFederate to mask individual extended attributes in its logs (see *Attribute masking* on page 123).

> **Tip:** If you are creating or updating a SAML or an OpenID Connect 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. 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 (see www.incommon.org/participants), the assertion may 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 (tools.ietf.org/html/rfc4524) and RFC4519 (tools.ietf.org/html/rfc4519). The following table describes a subset of the OIDs (object IDs) 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, refer to 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.

Results

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.

Manage target session mappings

About this task

Remote users arriving at your site via an SSO request do so in order to use specific applications or gain access to protected resources. Based on the nature of the business relationship and the agreement with your partner, you may be expected to provide access to these applications. Therefore, integration between your federation SP server and local applications is important. PingFederate uses SP adapters to identify the user to your application based on attributes sent in an SSO token. A configured adapter is known as an adapter instance.

In a basic scenario, you map an SP adapter instance to an IdP connection on the **Target Session Mapping** screen and complete its mapping configuration through a series of sub tasks. 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* on page 324.

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 which you map values from the SSO tokens.

ⓘ **Tip:** To learn more about federation hub, see *Federation hub use cases* on page 127.

On the **Target Session Mapping** screen (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), you should map multiple SP adapter instances. If you are bridging an identity provider to multiple service providers, you should map multiple authentication policy contracts.

Note that the **Target Session Mapping** configuration does not apply when the **No Mapping** option is chosen on the **Identity Mapping** screen.

Steps

- To map an SP adapter instance, click **Map New Adapter Instance**.
- To map an APC, click **Map New Authentication Policy**.
- To 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.
- To 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**.

Results

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 (see *Configure target URL mapping* on page 586).

Select a target session

About this task

The first task of the mapping configuration is to map an adapter instance or an authentication policy contract to your connection.

Steps

- To map an adapter instance, follow these steps:
  a. Select an adapter instance from the list.

     If you do not see the desired adapter instance, click **Manage Adapter Instances** to create a new instance of any deployed adapter.
  b. Select the **Override Instance Settings** check box if you want to customize one or more adapter settings for this connection alone.

     When selected, the administrative console adds a new set of sub tasks (the **Override Instance** screen 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 plug-in configurations* on page 119.
- To map an APC, select an adapter instance from the list.

  If you do not see the desired APC, click **Manage Authentication Policy Contracts** to create a new policy contract.

Results

If you are editing a currently mapped adapter 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 adapter settings specifically for this connection.

If you are editing a currently mapped APC, no changes can be made on this screen.

Override an SP adapter instance

About this task

On the **Override Instance** screen, 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 settings screens, select the **Override** check box, make your changes, and then click **Next**. When you are finished, click **Done** to continue with the rest of the mapping configuration.

  Note: The override setting screens are functionally identical to those used for creating a new adapter instance (see).

Results

If you are editing a currently mapped adapter instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You may also remove all overridden settings on a per-screen basis by clearing the **Override** check box near the top of the screen.

Restrict a target session to certain virtual server IDs

About this task

When you multiplex one connection for multiple environments (see), you can enforce integration requirements by restricting a target session to certain virtual server IDs on the **Virtual Server IDs** screen. By default, no restriction is imposed.

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 target session.

Results

If you are editing a currently mapped adapter instance or APC, you can toggle the **Restrict Virtual Server IDs** setting. You can also change the allowed virtual server IDs.

Choose an attribute mapping method

About this task

On the **Adapter Data Store** screen for SP adapter mapping (or the **Attribute Retrieval** screen for authentication policy contract mapping), you select if and how PingFederate should query a local data store to help fulfill the attribute contract in conjunction with attribute values from the SSO token.

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 data store can supply it.

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 configure data store queries as part of the fulfillment configuration for the applicable APC.

Tip: To learn more about authentication policies, see *Authentication policies* on page 324.

Steps

- If the attribute contract contains all the attributes that your application requires, select **Use only the attributes available in the SSO assertion**.
- To set up a data store query, select **Use the SSO assertion (or provider claims) to look up additional information** and then follow a series of sub tasks to complete the configuration.

  For step-by-step instructions, see *Choose a data store* on page 872.

Results

If you are editing a currently mapped adapter instance or APC, you can change the mapping method, which may require additional configuration changes in subsequent tasks.

Configure target session fulfillment

About this task

On the **Adapter Contract Fulfillment** screen, 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.

If you are bridging an identity provider to one or more service providers, the values mapped to the authentication policy contracts are used by the associated SP connections to create assertions for the service providers (see *Federation hub use cases* on page 127).

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

Steps

- 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**. 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. In addition, this source appears only if you have elected to use account linking for a target session on the **Identity Mapping** screen.
  - **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 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 is populated with the available context of the transaction. Select the desired context from the list.

  Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

  Note: If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
- To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

  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 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** as the source and enter a static value.

  This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

- **LDAP**, **JDBC**, or **Other**

  When selected, the **Value** list is populated with attributes that you have selected from the data store. Select the desired attribute from the list.
- **Expression** (when enabled)

  This option provides more complex mapping capabilities; for example, 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 (see **Text**).

  Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
- **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 data store, when applicable, using this syntax:

  `${ds.attribute}`

  where *attribute* is any attribute that you have selected from the data store.

  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 a variety of reasons why you might hard code a text value. For example, if your web application provides a consumer service, you might want to supply a particular promotion code for the partner.

Results

If you are editing a currently mapped adapter instance or APC, you can update the mapping configuration, which may require additional configuration changes in subsequent tasks.

Define issuance criteria for SP Browser SSO

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that 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 (see), 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 (see).

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. See the following table for more information.

Source	Description
AccountLink	Select to evaluate the **Local User ID** value of the user. Visible and applicable only if **Account Linking** is the selected identity mapping method (see).
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 data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.
4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.
5. Enter a custom error message under **Error Result**.

   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.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the target session mapping

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the session creation summary

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Manage protocol settings

About this task

The **Protocol Settings** screen provides the launching point for configuring partner endpoints, message customizations, and other protocol-specific settings for browser-based SSO connections.

SAML 2.0

- Outbound SSO bindings (POST, redirect, artifact) and the corresponding SSO service URLs
- Outbound 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 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 OP in its authorization and token requests
- The OpenID Connect login type and authentication scheme used by PingFederate when communicating with the OpenID Provider
- The authorization endpoint, the token endpoint, the user information (UserInfo) endpoint, and the JWKS URL
- Default target URL
- Authentication context mappings

Steps

- To continue, click **Configure Protocol Settings**.

Specify SSO service URLs (SAML)

About this task

The SSO service endpoint is a location to which PingFederate to send authentication requests when SSO is initiated at your site, according to partner requirements. It is applicable to all SAML versions when the SP-initiated SSO profile is enabled.

For SAML 2.0 connections, you associate bindings to the endpoints where your 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 may 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.
   a. Enter the SSO service endpoint to the **Endpoint URL** field.

      You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.

      For SAML 1.x connections, this is the only configurable item on the **SSO Service URL** screen.

      The remaining steps on the **SSO Service URLs** screen are only applicable to SAML 2.0 connections.
   b. Select a SAML binding from the list; for example, **POST**.
   c. Click **Add**.
   d. Optional: Repeat to add additional SSO service endpoints.
2. Optional: Customize messages using OGNL expressions.

   Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see *Attribute mapping expressions* on page 859. In addition, message customization is not applicable to SAML 1.x connection.

   a. Click **Show Advanced Customizations**.
   b. Select a message type from the list.
   c. Enter an OGNL expression to fulfill your use case.

      Note: For more information about **Message Type**, available variables, and sample OGNL expressions, see *Customize assertions and authentication requests* on page 864.
   d. Click **Add**.
   e. Optional: Repeat to add another message customization.

Specify a service URL (WS-Federation)

About this task

The service endpoint URL is a location to which PingFederate sends RST (Request for Security Token) and SLO messages.

To protect against session token hijacking, PingFederate provides an option to validate wreply for SLO. When the option is enabled, you can specify additional allowed domains and paths in this screen. 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: Settings to enter additional allowed domains and paths appear only if the option to validate wreply for SLO is enabled (see *Manage partner redirect validation* on page 319).

Steps

1. Enter the WS-Federation protocol endpoint at the IdP site in the **Endpoint URL** field.

   You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.

2. Optional: Specify additional allowed domains and paths.
   a. 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.
   b. Enter the expected domain name or IP address of this resource under **Valid Domain Name**.

      Enter a value without the protocol; for example: `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.
   c. Optional: Enter the exact path of this resource under **Valid Path**.

      Starts 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 may allow specific query parameter (or 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.
   d. Optional: 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.

Define SLO service URLs (SAML 2.0)

About this task

On the **SLO Service URLs** screen, you associate bindings to the endpoints where your IdP receives logout requests when SLO is initiated at your site and where PingFederate sends SLO responses when it receives SLO requests from the IdP.

This step applies only to SAML 2.0 connections when either SLO profile is selected on the **SAML Profiles** screen.

Steps

1. Select a SAML binding from the list; for example, **POST**.
2. Enter the SLO endpoint URL to the **Endpoint URL** field.

   You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.

3. Optional: Enter an URL to the **Response URL** field.

   When specified, it 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 may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.
4. Click **Add**.
5. Optional: Repeat to add additional SLO endpoints.

Results

If you are editing an existing connection, you can reconfigure the SLO endpoints, which may require additional configuration changes in subsequent tasks.

Select allowable SAML bindings (SAML)

About this task

On the **Allowable SAML Bindings** screen, you select the one or more bindings that your IdP partner can use to send SAML assertions or SAML 2.0 SLO messages.

This configuration applies to all SAML connections.

Steps

- Select only the applicable SAML binding (or bindings) based on your partner agreement.

  If you have specified an SSO or SLO endpoint using the artifact (outbound) binding, you must including SOAP as one of the allowable (inbound) binding.

Results

If you are editing an existing connection, you can reconfigure the allowable bindings, which may require additional configuration changes in subsequent tasks.

Specify an artifact lifetime (SAML 2.0)

About this task

When PingFederate sends an artifact to your IdP's SSO or SLO service endpoint, an element in the message indicates how long it should be considered valid. On the **Artifact Lifetime** screen, specify the expiry information in seconds.

You can change the default value per your requirements, if needed. Also consider synchronizing clocks between your server and your partner's SAML gateway server. If clocks are not synchronized, you might need to set the artifact lifetime to a higher value.

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).

Results

If you are editing an existing connection, you can update the artifact lifetime.

Define artifact resolver locations (SAML)

About this task

When the artifact binding is enabled is enabled as one of the allowable bindings on the **Allowable SAML Bindings** screen, you must provide an artifact resolution service (ARS) endpoint. This is the location where PingFederate sends back-channel requests to resolve artifacts received from the IdP.

SAML 2.0 connections allows multiple ARS endpoints. For SAML 1.x connection, you can only enter one ARS endpoint.

Steps

1. Enter an ARS endpoint.
   a. Enter the ARS endpoint URL.

      You may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen.

      If you are configuring a SAML 1.x connection, you can only enter one ARS endpoint on the **Artifact Resolver Location** screen.
   b. Optional: Enter an integer to the **Index** field for this ARS endpoint.

      (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.
   c. Click **Add**.
   d. Optional: Repeat to add additional ARS endpoints.

      (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 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.

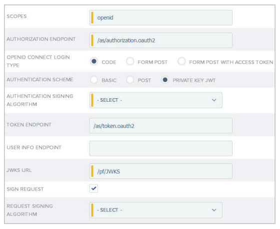
Results

If you are editing an existing connection, you may reconfigure any ARS endpoint (or the source ID value for SAML 1.x).

Configure OpenID Provider information

Steps

1. On the **OpenID Provider Info** screen, provide the scopes, the endpoints, and the authentication scheme; for example:



SCOPES	openid
AUTHORIZATION ENDPOINT	/as/authorization.oauth2
OPENID CONNECT LOGIN TYPE	[✓ CODE] FORM POST FORM POST WITH ACCESS TOKEN
AUTHENTICATION SCHEME	BASIC POST [✓ PRIVATE KEY JWT]
AUTHENTICATION SIGNING ALGORITHM	- SELECT -
TOKEN ENDPOINT	/as/token.oauth2
USER INFO ENDPOINT	
JWKS URL	/pf/JWKS
SIGN REQUEST	✓
REQUEST SIGNING ALGORITHM	- SELECT -

Note: If you have chosen to load the metadata from the OP on the **General Info** screen, 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 authentication and 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 may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen. 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**.

Field	Description
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 via HTTP Basic authentication. ▪ Select **POST** to submit credentials via POST. ▪ Select **Private Key JWT** to authenticate via the private_key_jwt Client Authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication). The default selection (without loading metadata from the OP) is **Basic**.
Authentication Signing Algorithm	Select the algorithm that PingFederate uses to sign the JWT. Applicable and visible only when **Private Key JWT** is the chosen authentication scheme. If PingFederate is either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . **Note:** Based on the chosen signing algorithm, PingFederate selects the signing JSON Web Key (JWK) from its JWK Set (JWKS) at runtime. In order for the OP to validate the signed JWT, ensure that the OP can access your PingFederate JWKS endpoint, which returns the current set of JSON Web Keys. The PingFederate JWKS endpoint is located at `<Base URL>/pf/JWKS`, where **Base URL** is defined on the **System# Protocol Settings# Federation Info** screen. 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 OpenID Provider configuration endpoint to obtain the information (see).

Field	Description
Token Endpoint, UserInfo Endpoint, and JWKS URL	Various 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 in order 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 11 runtime environment or integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see .) 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. This check box is not selected by default, in which case PingFederate sends request parameters via 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 either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . **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 the **System**# **Protocol Settings**# **Federation Info** screen. 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 (see).

2. Optional: Remain on the **OpenID Provider Info** screen and specify the request parameters that are allowed to be included in the authentication requests to the OP under **Request Parameters** (see).

Configure default target URLs

About this task

Use the **Protocol Settings**# **Overrides** screen to assign a default target URL for this IdP connection.

Steps

Optional: Enter a URL in the **Default Target URL** field.

If specified, the value overrides the default target setting defined on the **Service Provider**# **Default URLs** screen.

Note: The SAML 1.x specifications for IdP-initiated 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 on this screen will not be used for those transactions.

Override authentication context in an IdP connection

About this task

Authentication context values can be mapped between the local and remote values in an OpenID Connect or a SAML 2.0 IdP connection. 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 may 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. Select the applicable IdP connection from the **Service Provider** menu.

   If the applicable connection is not one of the most recently edited connections, click **Manage All** under **IdP Connections**, and then select it from the **IdP Connections** screen.

   You may also configure authentication context overrides when you create a new IdP connection.
2. On the **Activation & Summary** screen, scroll down to the **Protocol Settings** section, and then click **Overrides**.
3. On the **Overrides** screen, specify the **Local** and **Remote** entry, and then click **Add**.
4. Repeat the previous step to define additional mappings.

   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.
5. Click **Save** to complete the configuration.

   Alternatively, click **Next** to carry on with the rest of the connection settings.

Example

Example

Suppose you are the 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 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	AuthnContext values
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.

Configure signature policy

About this task

The **Signature Policy** screen provides options controlling how digital signatures are used for SAML and WS-Federation SSO messages. The choices made on this screen depend on your partner agreement (see).

Digital signing is required for SAML response messages sent from the IdP via 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 screen. Note that 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 SP may 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** screen.)

Steps

- To continue, select the option (or options) based on your partner agreement.

Results

If you are editing an existing connection, you can reconfigure the digital signature policy, which may require additional configuration changes in subsequent tasks.

Specify XML encryption policy (for SAML 2.0)

About this task

For SAML 2.0 configurations, in addition to using signed assertions to ensure authenticity, you and your partner may also agree to encrypt all or part of an assertion to improve privacy. If so, you can configure these settings on the **Encryption Policy** screen. There are a few options; each option provides different capabilities.

Option	Name identifier (SAML_SUBJEC	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.

To enable encryption:

Steps

1. Select the **Allow encrypted SAML Assertions and SLO messages** option.
2. Choose whether this 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 SLO messages, allow its encryption for inbound messages, or both.

Results

If you are editing an existing connection, you can reconfigure the XML encryption policy, which may require additional configuration changes in subsequent tasks.

Review protocol settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review Browser SSO settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Manage Attribute Query profile

At the Attribute Query step you configure your connection to request user attributes from your partner IdP, if you have chosen this option (see). Attribute queries are not dependent on single sign-on but may be used independently or in conjunction with Browser SSO or provisioning to provide flexibility in how a user authenticates with SP applications (see and the */sp/startAttributeQuery.ping* on page 765 SP application endpoint).

Set the Attribute Authority Service URL

About this task

Attribute Authority is the term used to refer to an IdP that provides user attributes to an *Attribute Requester* (your SP site). The **Attribute Authority Service URL** corresponds to the endpoint location where Attribute Query requests are received by your IdP partner (see *Attribute Query and XASP* on page 45).

Steps

- Enter the fully qualified URL or a relative path if you have defined a base URL on the **General Info** screen (see *Identify the partner* on page 599).

Map attribute names for Attribute Query

About this task

If the application at your site uses different names for user attributes than the names defined by the Attribute Authority, then you need to map them on this screen. When the 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.

This information must be predetermined in your agreement with this connection partner.

Steps

- To map an attribute, configure the **Local Name** and **Remote Name** fields for the attribute and then click **Add**.
- To modify an attribute name, follow these steps:
  a. Click **Edit** under **Action** for the attribute.
  b. 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.

Configure security policy for Attribute Query

About this task

This screen allows you to specify the digital signing and encryption policy to which you and your partner have agreed. These selections will trigger requirements for setting up Credentials (see).

This screen also allows you to mask incoming attribute values in log files (see *Attribute masking* on page 123). When you enable this selection, all user attributes returned from this IdP are masked.

Steps

- Select or clear the check boxes the relevant check boxes.

Review the Attribute Query settings

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

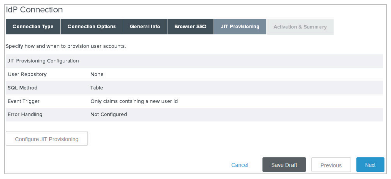
  Tip: When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.
- To discard your changes, click **Cancel**.

Configure just-in-time provisioning

About this task

PingFederate's just-in-time (JIT) provisioning allows SPs to create user accounts on the fly during SSO events, based on attributes received in SSO tokens from IdPs. An SP can also use the feature to update existing user records.

> **Note:** This configuration task is presented in the administrative console only when the **JIT Provisioning** check box is selected in the **Connection Options** screen.



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

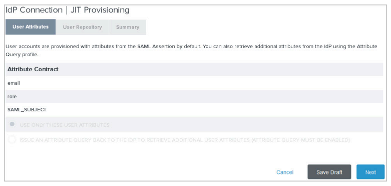
Steps

1. Create a new IdP connection or select an existing IdP connection on the **Service Provider** menu.
2. On the **Connection Type** screen, select the **Browser SSO Profiles** check box and a protocol from the list.
3. On the **Connection Options** screen, select the **Browser SSO** check box and then the **JIT Provisioning** check box.
4. Complete the **Browser SSO** configuration.
5. On the the **JIT Provisioning** screen, click **Configure User Provisioning** to begin the configuration of JIT provisioning.

Select attribute sources (SAML 2.0)

About this task

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 IdP in a follow-on Attribute Query transaction. The **User Attributes** screen displays the attributes expected in the assertion from this IdP.



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:** Attribute Query is a SAML 2.0 profile. For OpenID Connect, SAML 1.x, and WS-Federation connections, this screen 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 screen.

  You configure the Attribute Query profile later in the task flow.

Identify the user repository

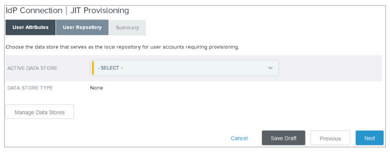
About this task

PingFederate's JIT Provisioning supports several directory servers and Microsoft SQL Server (see *System requirements* on page 60).

Note: PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see .

Steps

- On the **User Repository** screen, select a data store 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 -

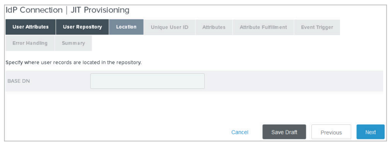
  DATA STORE TYPE: None

  Tip: If the desired data store is not shown in the list, then PingFederate has not yet been configured to access it. Click **Manage Data Stores** to configure your data store.
- If you are using an LDAP store, refer to the sections immediately following:
  - *Specify an LDAP user-record location* on page 628
  - *Enter an LDAP filter* on page 629
  - *Identify provisioning attributes for LDAP* on page 629
- If you are using a Microsoft SQL Server, skip to this section:
  - *Choose a SQL method*

Specify an LDAP user-record location

About this task

After choosing a data store, indicate where in the store PingFederate should write new user records or update existing ones. Start by specifying where user records are located in your data store.



IdP Connection | JIT Provisioning

Specify where user records are located in the repository.

BASE DN:

Note: This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

Steps

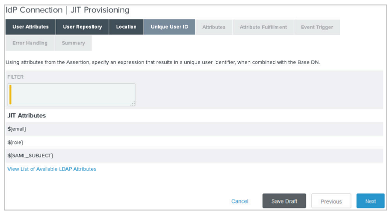
- Enter the base DN in the text field.

  A base distinguished name (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.

Enter an LDAP filter

About this task

On the **Unique User ID** screen, create an LDAP filter to identify user accounts to be provisioned (or updated) during SSO events. PingFederate uses this expression in conjunction with the Base DN value defined on the **Location** screen to locate existing account records and to add new ones.



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}

View List of Available LDAP Attributes

> **Note:** This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

Steps

- Enter the statement in the **Filter** text 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-data store. Click the link near the lower-left corner of the screen to see a list of available attributes.

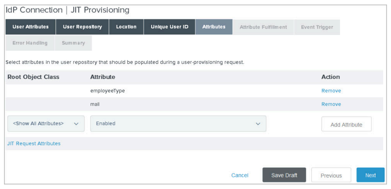
  The right side of the filter generally uses one or more attribute values passed in from the SSO token. Variables for these attributes, including the correct syntax, are listed under **JIT Attributes**.

  > **Tip:** If you are unfamiliar with writing LDAP queries, please refer to the documentation accompanying your LDAP installation.

Identify provisioning attributes for LDAP

About this task

On the **Attributes** screen, select the data-store attributes to be provisioned.



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

JIT Request Attributes

> **Note:** This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

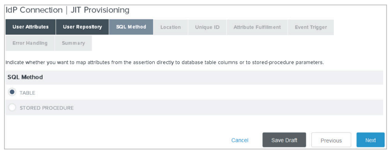
Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Choose a SQL method

About this task

For JDBC data stores, PingFederate allows you to map attributes directly to a single database table (the default) or to SQL stored-procedure parameters.



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 screen appears only when a Microsoft SQL Server data store is chosen on the **User Repository** screen.

Steps

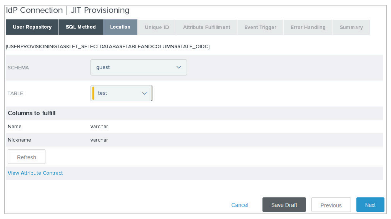
- Make a selection as needed and click **Next**.

  Depending on the selection, different steps appear under the **JIT Provisioning** task. Refer to the sections indicated below for more information:
- If you are mapping attributes directly to a table, refer to the topics sections immediately following:
  - *Specify a database user-record location* on page 630
  - *Specify a unique-ID database column* on page 631
- If you are using a stored procedure, skip to this section:
  - *Specify a stored-procedure location* on page 631

Specify a database user-record location

About this task

For database provisioning to a table, indicate where PingFederate should write new user records or update existing ones.



[USERPROVISIONINGTASKLET_SELECTDATABASETABLEANDCOLUMNSSTATE_OIDC]

SCHEMA: guest

TABLE: test

Columns to fulfill

Name	varchar
Nickname	varchar

Note: This screen appears only when a Microsoft SQL Server data store is chosen on the **User Repository** screen and the **Table** option is selected on the **SQL Method** screen.

Steps

- 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.

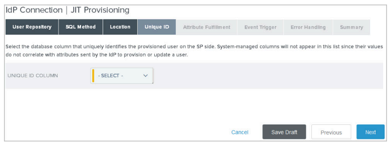
Field	Description
Columns to fulfill	For the selected table, all attributes and their data types are displayed. If the list of columns is not or might not be current (due to any recent changes), click **Refresh**. Later on the **Attribute Fulfillment** screen, all attributes must be mapped for the database insertion to succeed, although a null entry may be used for optional attributes.

Tip: Click the link near the lower-left corner of the screen to see a list of available attributes from the SSO token.

Specify a unique-ID database column

About this task

PingFederate uses the column specified on this screen to check whether a user record already exists for the incoming SSO token.


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 screen appears only when a Microsoft SQL Server data store is chosen on the **User Repository** screen and the **Table** option is selected on the **SQL Method** screen.

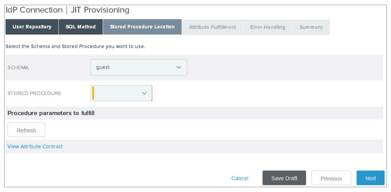
Steps

- Select a column that represents a unique characteristic about the database entry for a particular user.

Specify a stored-procedure location

About this task

If you are using a stored procedure for provisioning the user database, specify its location on this screen.


IdP Connection | JIT Provisioning

Select the Schema and Stored Procedure you want to use.

SCHEMA: guest

STORED PROCEDURE:

Procedure parameters to fulfill

Note: This screen appears only when a Microsoft SQL Server data store is chosen on the **User Repository** screen and the **Stored Procedure** option is selected on the **SQL Method** screen.

Important: The database account used by PingFederate must have access to the schema in which the stored procedure is located and to execute permission for the procedure.

Steps

- Select the database schema and the stored procedure.

Field	Description
Schema	Select the table structure that contains stored procedure within the database.
Stored Procedure	Select the stored procedure needed to provision the user database.
Procedure parameters to fulfill	For the selected procedure, all parameters and their data types are displayed. If the list of parameters is not or might not be current (due to any recent procedure revisions), click **Refresh**. You map attribute values from the SSO token to the parameters on the **Attribute Fulfillment** screen.

Tip: Click the link near the lower-left corner of the screen to see a list of available attributes from the SSO token.

Map attributes to a user account

About this task

The **Attribute Fulfillment** screen provides a means of mapping the 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. Besides values obtained from the SSO token, you may also map from the context of the SSO token, text with or without references values from the SSO token, and expression (if enabled).

If a Microsoft SQL Server data store is chosen on the **User Repository** screen, the **Attribute Fulfillment** screen, you may also 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 may use a PingFederate Java conversion method via OGNL expressions.

Steps

1. Select a source from the list for each target attribute or parameter.
   - Assertion or Provider Claims

     Values are contained in the SSO token from this IdP. When you make this selection, the associated **Value** list is populated by the attribute contract).
   - Context

     Values are returned from the context of the transaction at runtime.

     > **Note:** The HTTP Request is retrieved as a Java objects 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.)
   - Attribute Query

     This choice appears only if you have chosen to use the Attribute Query profile for provisioning.

     To map an attribute-query value, use this syntax: `${query_attribute}`

     You can also 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:** If you have not already done so, you may enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory. 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 on the **System# Cluster Management** screen, and restart all nodes.

     This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries (see below) are also available for expressions.

     > **Tip:** If multiple attribute values from one or multiple sources need to be mapped 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**

for such attributes, click **Datetime OGNL Examples** under the text box for syntax information and examples.

- System Managed (if applicable)

  This mapping option appears only when any automatically assigned attributes are among columns to be provisioned; for example, 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.

  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 text box 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 text box to explicitly set `null` in the column.

2. Select (or enter) an attribute value.

   All values must be mapped. However, for optional table columns, you may leave a text box blank (or, for string data types, enter `null` to avoid empty strings).

   Note that no value is required for **System Managed** attributes.

   Note: For Active Directory, enter `user` in the text box for objectClass. For the Oracle Directory Server, enter `inetOrgPerson`.
3. Optional: When mapping to a Microsoft SQL Server data store, test the insertion.

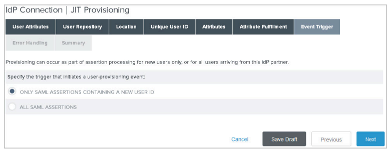
Table	**a.** Click **Test insert into '*table*'**. **b.** Enter values for each applicable target parameter. **c.** Click **Test Insert**. If the test succeeds, a confirmation is displayed along with the values inserted. **CAUTION:** Unless you wish to keep the test values in the database, click **Roll Back All Test Inserts**.
Stored procedure	**a.** Click **Test call to '*procedure*'**. **b.** Enter values for each applicable target parameter. **c.** Click **Test Stored Procedure Call**. For stored procedures, only a confirmation is displayed 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, if needed, must be handled manually.

When finished, click **Return to Attribute Fulfillment**.

Choose an event trigger

About this task

On the **Event Trigger** screen, choose whether PingFederate initiates user provisioning only when the user identifier is new or every time your site receives an SSO token. In the latter case (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

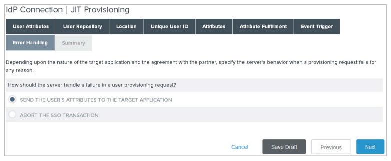
> **Note:** This screen does not appear for a Microsoft SQL Server data store if provisioning is accomplished using a stored procedure, because the procedure is always called for all SSO tokens. The procedure should handle both provisioning new users and updating existing ones (if desired).

Configure an error handling method

About this task

If user provisioning fails for any reason during SSO events, you can choose to stop the SSO or continue the process by passing the attributes on to your target application.

When SSO is aborted, the user is redirected to an error page and the failure is written to the server log.


IdP Connection | JIT Provisioning

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

Review the JIT provisioning configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure SCIM inbound provisioning

About this task

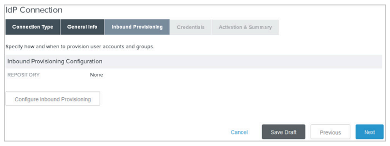
This configuration provides for two-way mapping of attributes. The first facilitates SCIM operations used to create and update records in the data store (see *Write user information to the data store* on page 641). The second allows the same SCIM client to retrieve those records and have the attribute values mapped back to their corresponding designation in the client store (see *Configure a SCIM response* on page 644).

The dual mapping is intended to provide greater flexibility, especially when needed for OGNL-expression transformations; for example, converting two attributes into one multivalued attribute and then back again.

Note: SCIM-client requests must include authentication credentials, which you configure later on the **Credentials# Back-Channel Authentication** screen. The same credentials needed for SSO or other types of transactions enabled as part of this IdP connection, if configured, are also used for SCIM transactions.

Steps

1. If you have not already done so, enable the **Inbound Provisioning** option (under the Server Provider role) on the **System# Protocol Settings# Roles & Protocols** screen.
2. Create a new IdP connection or select an existing IdP connection on the **Service Provider** menu.
3. On the **Connection Type** screen, select the **Inbound Provisioning** check box and one of these two options: **User Support** or **User and Group Support**.
4. On the **Inbound Provisioning** screen, click **Configure Inbound Provisioning** to begin the configuration of SCIM inbound provisioning.


IdP Connection

Specify how and when to provision user accounts and groups.

Inbound Provisioning Configuration

REPOSITORY None

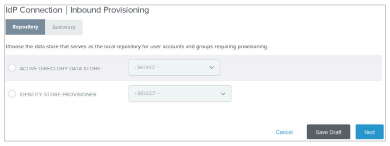
Specify the user repository

About this task

PingFederate currently supports AD user stores (requires an LDAPS connection to the data store) and custom identity store provisioners for inbound provisioning.

Steps

- Choose either the **Active Directory Data Store** or **Identity Store Provisioner** option and then specify the data store from the list.

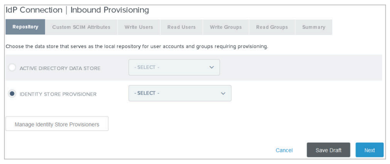

IdP Connection | Inbound Provisioning

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 data store is not shown in the list, then PingFederate is not yet configured to access the store. Click **Manage ...** to set up the desired data store.


IdP Connection | Inbound Provisioning

Choose the data store that serves as the local repository for user accounts and groups requiring provisioning.

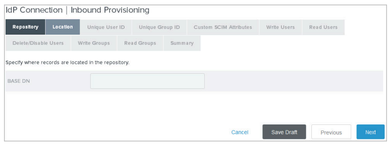
ACTIVE DIRECTORY DATA STORE - SELECT -

[✓ IDENTITY STORE PROVISIONER] - SELECT -

Identify an LDAP user-record location

About this task

After choosing a data store, indicate where in the data store user and group records exist so PingFederate can create, read, update, or delete/disable them.



Specify where records are located in the repository.

BASE DN

> **Note:** This screen appears only if you are configuring an LDAP user store for provisioning.

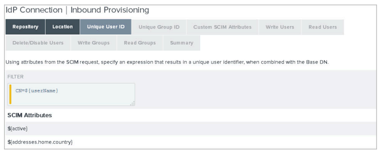
Steps

- Enter the base distinguished name of the tree structure where user records are stored in the **Base DN** field. PingFederate looks only at this node level or below it for user accounts that need provisioning.

Define a unique ID

About this task

On the **Unique ID** screen, create an LDAP filter to resolve user accounts for SCIM operations. PingFederate uses this expression in conjunction with the **Base DN** value (defined on the **Location** screen) 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 screen appears only if you are configuring an LDAP user store for provisioning.

Steps

- Enter the statement in the **Filter** text 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-data store. Click the link near the lower-left corner of the screen to see a list of available attributes.

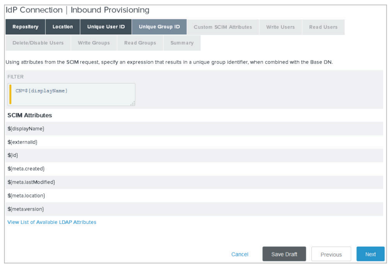
  The right side of the filter generally uses one or more attribute values passed in from the SCIM request. Variables for these attributes, including the correct syntax, are listed under **SCIM Attributes**.

  > **Tip:** If you are unfamiliar with writing LDAP queries, please refer to the documentation accompanying your LDAP installation.

Define a unique group ID

About this task

On the **Unique Group ID** screen, create an LDAP filter to resolve groups for SCIM operations. PingFederate uses this expression in conjunction with the **Base DN** value (defined on the **Location** screen) 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}

View List of Available LDAP Attributes

> **Note:** This screen appears only if you are configuring an LDAP user store for provisioning and the **User and Group Support** option is selected on the **Connection Type** screen.

Steps

- Enter the statement in the **Filter** text 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-data store. Click the link near the lower-left corner of the screen to see a list of available attributes.

  The right side of the filter generally uses one or more attribute values passed in from the SCIM request. Variables for these attributes, including the correct syntax, are listed under **SCIM Attributes**.

  > **Tip:** If you are unfamiliar with writing LDAP queries, please refer to the documentation accompanying your LDAP installation.

Define custom SCIM attributes

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** screen.

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 attributes

The 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.

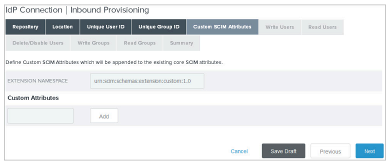
```
{
  "userName":"CBrown",
  "active":true,
  "schemas":[
    "urn:scim:schemas:core:1.0",
    "urn:scim:schemas:extension:custom:1.0"
  ],
  ...
  "urn:scim:schemas:extension:custom:1.0":{
    "supervisor":"JSmith",
    "territories":[
      "Montana",
      "Idaho",
      "Wyoming"
    ],
    "options":{
      "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

- To specify a schema extension, enter the URI of the schema extension in the **Extension Namespace** field.


IdP Connection | Inbound Provisioning

Repository | Location | Unique User ID | Unique Group ID | Custom SCIM Attributes | Write Users | Read Users | Delete/Disable Users | Write Groups | Read Groups | Summary

Define Custom SCIM Attributes which will be appended to the existing core SCIM attributes.

EXTENSION NAMESPACE: urn:scim:schemas:extension:custom:1.0

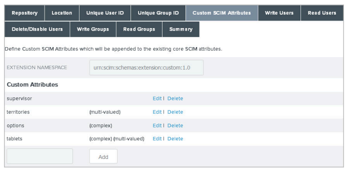
Custom Attributes

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.

  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		
supervisor		Edit | Delete
territories	(multi-valued)	Edit | Delete
options	(complex)	Edit | Delete
tablets	(complex) (multi-valued)	Edit | Delete
	Add	

  The administrative console displays the **Custom SCIM Attribute Options** screen, where you may

  - 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 *Configure custom SCIM attribute options* on page 640.)

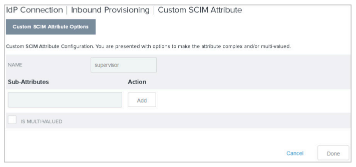
Configure custom SCIM attribute options

About this task

For the chosen custom attribute, use the **Custom SCIM Attribute Options** screen to 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

- 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

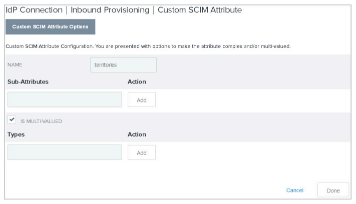
  Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

  NAME supervisor

Sub-Attributes	Action
	Add

  ☐ IS MULTI-VALUED

- To define the custom attribute as a simple multivalued attribute, select the **Is Multivalued** check box 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 territories

Sub-Attributes	Action
	Add

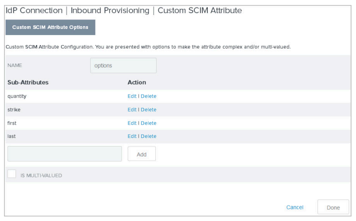
  ✓ IS MULTI-VALUED

Types	Action
	Add

- To define the custom attribute as a complex attribute, enter a sub-attribute and click **Add**.

  > **Tip:** Use the **Edit**/**Update**/**Cancel** links to make or undo a change to the name of a sub-attribute. Use the **Delete**/**Undelete** links to remove a sub-attribute or cancel the deletion.

  Repeat this step to add more sub-attributes as needed 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: options

Sub-Attributes	Action
quantity	Edit I Delete
strike	Edit I Delete
first	Edit I Delete
last	Edit I Delete

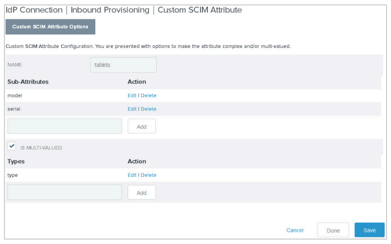
  IS MULTI-VALUED: ☐

- To define the custom attribute as a complex multivalued attribute:

  1. Enter a sub-attribute and click **Add**.

     > **Tip:** Use the **Edit**/**Update**/**Cancel** links to make or undo a change to the name of a sub-attribute. Use the **Delete**/**Undelete** links to remove a sub-attribute or cancel the deletion.

     Repeat this step to add more sub-attributes as needed.
  2. Select the **Is Multivalued** check box.
  3. If you have chosen Active Directory as your user store (see *Specify the user repository* on page 636), you must specify at least one value under the **Types** column for `type`, a reserved sub-attribute for a complex multivalued attribute.
  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 I Delete
serial	Edit I Delete

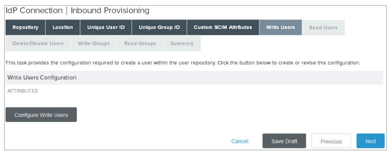
     IS MULTI-VALUED: ✓

Types	Action
type	Edit I Delete

Write user information to the data store

About this task

To configure how PingFederate completes create and update operations for user accounts from a SCIM request, identify incoming attributes and map them to data-store attributes.


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

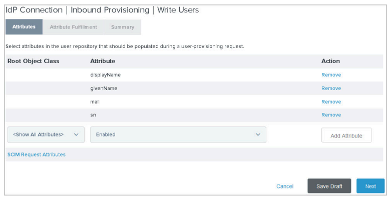
Steps

- Click **Configure Write Users** to continue.

Identify inbound provisioning attributes for LDAP

About this task

On the **Attributes** screen, select the data-store attributes you want to provision.



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

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

Several attributes are managed internally by PingFederate and do not require mapping:

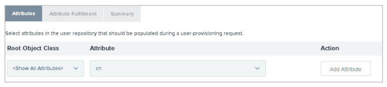
- objectClass
- unicodePwd
- objectGUID
- userAccountControl

You can override the internal management of objectClass and unicodePwd by selecting these attributes and mapping them to SCIM attributes on the **Attribute Fulfillment** screen. 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.

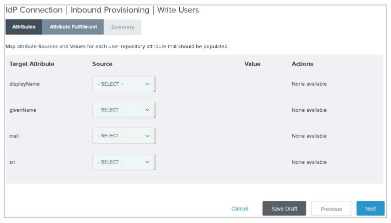
Root Object Class	Attribute	Action
<Show All Attributes>	cn	Add Attribute

2. Repeat the previous step for each attribute requiring provisioning.

Map attributes to user accounts

About this task

Use this screen to map attribute values in the SCIM request to user-account attributes.



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. Select a source from the list for each target attribute.

   - **Context**

     When selected, the **Value** list is populated with the available context of the transaction. Select the desired context from the list.

     > ⓘ **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

     > ⓘ **Note:** If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
     > - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.
     >
     >   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 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** as the source and enter a static value.
     >
     >   This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

   - Expression

     > ⓘ **Tip:** If you have not already done so, you may enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory. 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 on the **System**# **Cluster Management** screen, and restart all nodes.

This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries (see below) 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 drop-down list is populated by defined components of the SCIM request.
- 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.

2. Select (or enter) an attribute value.

   All target attributes must be mapped.
3. Click **Done**.

Review user mapping (Write Users) configuration

About this task

The **Summary** screen provides an overview of the inbound provisioning configuration for request mapping.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

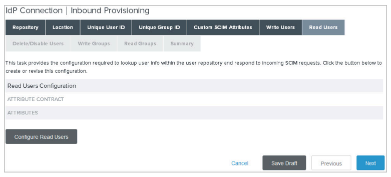
> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure a SCIM response

About this task

To configure a SCIM response to a request to read and return provisioned SCIM attributes, identify and map the user account attributes you want to include.


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

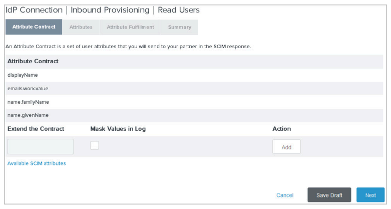
Steps

- Click **Configure Read Users** to continue.

Identify expected user attributes for the SCIM response

About this task

An attribute contract is a set of user attributes that you and your partner have agreed will be sent in a SCIM response for this connection. The attributes you mapped to user account attributes in the **Write Users** flow 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

Click **Available SCIM Attributes** near the lower-left corner of the screen 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
- active

Steps

- 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**.

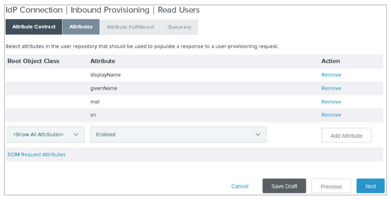
  Attribute names are case-sensitive and must correspond to the attribute names expected by your partner. To see a list of available attributes, click **Available SCIM attributes**.
- To modify an attribute name or masking selection, click **Edit** under **Action** for the attribute, make the change, and click **Update**.

  Note: If you change your mind, ensure that you click **Cancel** under **Actions**, not the **Cancel** button, which discards any other changes you might have made in the configuration steps.
- To delete an attribute, click **Delete** under **Action** for the attribute.

Identify LDAP attributes for the SCIM response

About this task

Select the LDAP attributes you want to map to attributes in the SCIM response.


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
<Show All Attributes>	Enabled	Add Attribute

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

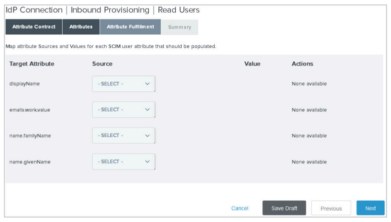
Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Map attributes into the SCIM response

About this task

Use this screen to map outgoing user-account attributes to SCIM responses to READ requests.


IdP Connection | Inbound Provisioning | Read Users

Map attribute Sources and Values for each SCIM user attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available
emails.work.value	- SELECT -		None available
name.familyName	- SELECT -		None available
name.givenName	- SELECT -		None available

Steps

1. Select a source from the list for each target attribute.
   - **Context**

     When selected, the **Value** list is populated with the available context of the transaction. Select the desired context from the list.

     Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

     Note: If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
     - To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

       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 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** as the source and enter a static value.

       This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.
   - Expression

     Tip: If you have not already done so, you may enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/`

`pingfederate/server/default/data/config-store` directory. 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 on the **System# Cluster Management** screen, and restart all nodes.

This option provides more complex mapping capabilities; for example, transforming outgoing values into different formats. All of the variables available for text entries (see below) 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 is populated by the LDAP attributes you identified for this data store.
- Identity Store

  Values are returned from your query. When you make this selection, the Value list is populated by the Identity Store attributes you identified for this data store.
- 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.

2. Select (or enter) an attribute value.

   All target attributes must be mapped.
3. Click **Done**.

Review SCIM response (Read Users) configuration

About this task

The **Summary** screen provides an overview of the inbound provisioning configuration for SCIM response mapping.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

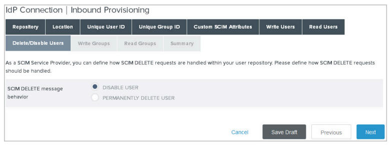
  Tip: When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.
- To discard your changes, click **Cancel**.

Configure the handling of SCIM delete requests

About this task

Use this screen to define how SCIM delete requests are handled within your user data store.

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 data store.



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 screen appears only if you are configuring an LDAP user store for provisioning.

Steps

- Select **Disable User** to make the user inactive within the data store. This approach might be preferred in situations where accounts must be retained for auditing reasons.

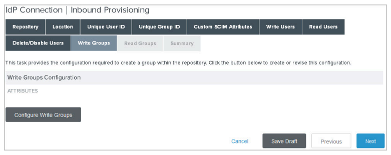
  In order to be SCIM compliant when deleting users, PingFederate returns an HTTP 404 response code for all subsequent operations related to the user-effectively treating the user as if they have been deleted from the LDAP user store (see the *SCIM specifications*).

  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.
- Select **Permanently Delete User** to remove the user from the data store.

Write group information to the data store

About this task

To configure how PingFederate completes create and update operations for groups from a SCIM request, identify incoming attributes and map them to data-store attributes.



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

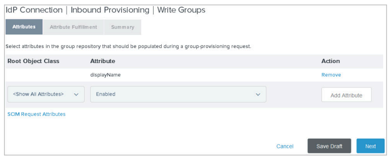
Steps

- Click **Configure Write Groups** to continue.

Identify inbound provisioning group attributes for LDAP

About this task

On the **Attributes** screen, select the data-store attributes you want to provision.



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 screen appears only if you are configuring an LDAP user store for provisioning and the **User and Group Support** option is selected on the **Connection Type** screen.

Several attributes are managed internally by PingFederate and do not require mapping:

- objectClass
- objectGUID
- member

You can override the internal management of objectClass by selecting and mapping it to a SCIM attribute on the **Attribute Fulfillment** screen. In this case, the values you supply are used. The objectGUID and member attributes cannot be overridden and are ignored if selected.

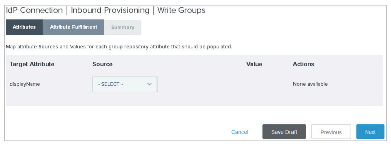
Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Map attributes to groups

About this task

Use this screen to map attribute values in the SCIM request to group attributes.



IdP Connection | Inbound Provisioning | Write Groups

Map attribute Sources and Values for each group repository attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available

Steps

1. Select a source from the list for each target attribute.

   - **Context**

     When selected, the **Value** list is populated with the available context of the transaction. Select the desired context from the list.

     Note: The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

     Note: If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
- To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

  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 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** as the source and enter a static value.

  This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

- Expression

  Tip: If you have not already done so, you may enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory. 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 on the **System**# **Cluster Management** screen, and restart all nodes.

  This option provides more complex mapping capabilities; for example, transforming incoming values into different formats. All of the variables available for text entries (see below) 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 Group

  When you make this selection, the associated Value drop-down list is populated by defined components of the SCIM request.
- 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.

2. Select (or enter) an attribute value.

   All target attributes must be mapped.
3. Click **Done**.

Review group mapping (Write Groups) configuration

About this task

The **Summary** screen provides an overview of the inbound provisioning configuration for request mapping.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and continue with the rest of the configuration.

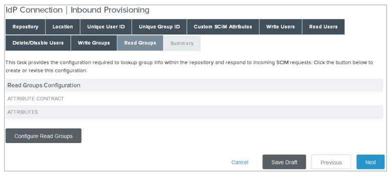
> **Tip:** When editing an existing configuration, you may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure a SCIM response for groups

About this task

To configure a SCIM response to a request to read and return provisioned SCIM attributes, identify and map the group attributes you want to include.



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.

Read Groups Configuration

ATTRIBUTE CONTRACT

ATTRIBUTES

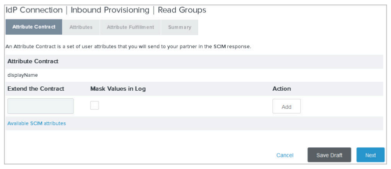
Steps

- Click **Configure Read Groups** to continue.

Identify expected group attributes for the SCIM response

About this task

An attribute contract is a set of group attributes that you and your partner have agreed will be sent in a SCIM response for this connection. The attributes you mapped to group attributes in the Write Groups flow appear at the top of the screen.


IdP Connection | Inbound Provisioning | Read Groups
Attribute Contract
Attributes
Attribute Fulfillment
Summary
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
Cancel
Save Draft
Next

Click **Available SCIM Attributes** near the lower-left corner of the screen 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

- 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**.

  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**.

- To modify an attribute name or masking selection, click **Edit** under **Action** for the attribute, make the change, and click **Update**.

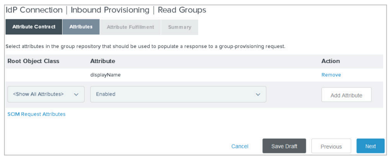
> **Note:** If you change your mind, ensure that you click **Cancel** under **Actions**, not the **Cancel** button, which discards any other changes you might have made in the configuration steps.

- To delete an attribute, click **Delete** under **Action** for the attribute.

Identify LDAP group attributes for the SCIM response

About this task

Select the LDAP attributes you want to map to attributes in the SCIM response.


IdP Connection | Inbound Provisioning | Read Groups

Attribute Contract | Attributes | Attribute Fulfillment | Summary

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

SCIM Request Attributes

Cancel | Save Draft | Previous | Next

> **Note:** This screen appears only if you are configuring an LDAP user store for provisioning and the **User and Group Support** option is selected on the **Connection Type** screen.

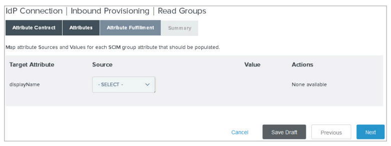
Steps

1. Select a root object class and an attribute from the lists, then click **Add Attribute**.
2. Repeat the previous step for each attribute requiring provisioning.

Map group attributes into SCIM response

About this task

Use this screen to map outgoing group attributes to SCIM responses to READ requests.


IdP Connection | Inbound Provisioning | Read Groups

Attribute Contract | Attributes | Attribute Fulfillment | Summary

Map attribute Sources and Values for each SCIM group attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available

Cancel | Save Draft | Previous | Next

Steps

1. Select a source from the list for each target attribute.
   - **Context**

     When selected, the **Value** list is populated with the available context of the transaction. Select the desired context from the list.

     > **Note:** The **HTTP Request** context value is retrieved as a Java object rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values.

     > **Note:** If you are configuring an **OAuth Attribute Mapping** configuration and `PERSISTENT_GRANT_LIFETIME` has been added as an extended attribute on the **OAuth Server# Authorization Server Settings** screen, you have the option to 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** as the source and **Default Persistent Grant Lifetime** as the value.
- To set lifetime based on the outcome of attribute mapping expressions, select **Expression** as the source and enter an OGNL expression as the value.

  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 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** as the source and enter a static value.

  This is most suitable for testing purposes or use cases where the persistent grant lifetime must always be set to a certain value in some specific grant-mapping configurations.

- Expression

  Tip: If you have not already done so, you may enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory. 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 on the **System**# **Cluster Management** screen, and restart all nodes.

  This option provides more complex mapping capabilities; for example, transforming outgoing values into different formats. All of the variables available for text entries (see below) 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 is populated by the LDAP attributes you identified for this data store.
- Identity Store

  Values are returned from your query. When you make this selection, the Value list is populated by the Identity Store attributes you identified for this data store.
- 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.

2. Select (or enter) an attribute value.

   All target attributes must be mapped.
3. Click **Done**.

Review SCIM response for groups (Read Groups) configuration

About this task

The **Summary** screen provides an overview of the inbound provisioning configuration for SCIM response mapping.

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the inbound provisioning configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Configure security credentials

About this task

The **Credentials** screen 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**.

  Refer to subsequent topics for configuration steps.

Manage back-channel authentication

About this task

When you configure 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 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.

Steps

- Refer to subsequent topics for configuration steps.

Configure back-channel authentication for outbound messages

About this task

Steps

1. On the **Back-Channel Authentication** screen, click **Configure** to the right of the list of messages under **Send to your partner**.
2. On the **Outbound SOAP Authentication Type** screen, 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)** screen.

   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** screen. If you have not yet created or imported the client certificate, click **Manage Certificates** to do so (see).

   > **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 a subsequent screen, **Digital Signature Settings**.

   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 may be used in any combination or independently.
3. On the **Summary** screen, review your configuration and perform one of the following tasks:

   Amend your configuration

   Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Configure back-channel authentication for inbound messages

About this task

Steps

1. On the **Back-Channel Authentication** screen, click **Configure** to the right of the list of messages under **Received from your partner**.
2. On the **Inbound Authentication Type** screen, 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)** screen.

   > **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 screens (see next step).

   Digital Signature (Browser SSO profile only)

   You select a signing certificate on a subsequent screen, **Signature Verification Settings**.

   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 may 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 may 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** screen.

   Anchored

   The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you may 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 **Security**# **Trusted CAs** screen.

   Unanchored

The partner certificate is self-signed or you want to trust a specified certificate.

> **Note:** When anchored certificates are used between partners, certificates may be changed without sending the update to your partner. If the certificate is unanchored, any changes must be promulgated.
>
> (For more information, see .)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: **a.** Enter the Subject DN of the certificate. **b.** (Optional) 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** screen, 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 (see).

4. On the **Summary** screen, review your configuration and perform one of the following tasks:

Amend your configuration

Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Manage digital signature settings

About this task

This step defines the private key you will use to sign SSO authentication or attribute requests (optionally) or SAML 2.0 SLO messages for this IdP. In addition, the step allows you to include "Key Info" with the XML message if you and your partner have agreed to this option.

Digital signing applies to SP-initiated SSO under SAML 2.0, when specified by your partner agreement, and to either SLO profile using the POST or redirect bindings. The step 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. Select a signing certificate from the list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see).
2. 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.
3. Optional: 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.

Manage signature verification settings

About this task

Under SAML 2.0 specifications, when your site receives any SAML 2.0 messages via the POST or Redirect bindings, the messages must be digitally signed. 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 IdP, SSO assertions, SAML 2.0 artifacts, or SOAP messages might also require signatures.

Steps

1. On the **Signature Verification Settings** screen, click **Manage Signature Verification Settings**.
2. On the **Trust Model** screen, select a trust model on the **Certificate Verification Method** screen.

   Anchored

   The partner certificate must be signed by a trusted certificate authority (CA). Optionally, you may 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 **Security# Trusted CAs** screen.

   Important: If you are using the redirect binding for 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 may 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* on page 117.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: **a.** Enter the Subject DN of the certificate or extract it from your SP partner's certificate if the certificate is stored on an accessible file system. **b.** (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.
Unanchored	On the **Signature Verification Certificate** screen: **a.** Select a certificate from the list. If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so (see *Manage certificates from partners* on page 310). **b.** (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 situation 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** screen, review your configuration and perform one of the following tasks:

Amend your configuration

Click the corresponding screen 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

Discard your changes

Click **Cancel**.

Choose an encryption certificate (SAML 2.0)

About this task

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 SLO requests from your site may also be encrypted (if the connection uses SP-initiated SLO). For more information, see *Specify XML encryption policy (for SAML 2.0)* on page 624.

You must also choose a certificate if encryption of the name identifier is required for an Attribute Request profile.

Steps

1. Optional: Select an option under **Block Encryption Algorithm**.

   > **Important:** Due to the import restrictions of some countries, Oracle Server JRE (Java SE Runtime Environment) 8 has built-in restrictions on available cryptographic strength (key size). To use larger key sizes, the Java Cryptography Extension (JCE) "unlimited strength" jurisdiction policy must be enabled. For more information, see the Java 8 release notes from Oracle (www.oracle.com/technetwork/java/javase/8u151-relnotes-3850493.html).
   >
   > For Oracle Java SE Development Kit 11, the JCE jurisdiction policy defaults to unlimited strength. For more information, see the Oracle JDK Migration Guide (docs.oracle.com/en/java/javase/11/migrate/).

   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 (www.w3.org/TR/xmlenc-core/).
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, we recommend upgrading such connections to use the newer algorithm **RSA-OAEP**.

   The default selection is **RSA-OAEP**.
3. Select a partner certificate from the list.

   If you have not yet imported the certificate from your partner, click **Manage Certificates** to do so (see).

Choose a decryption key (SAML 2.0)

About this task

As part of XML encryption, you must identify a certificate and key for PingFederate to use to decrypt incoming assertions or assertion elements (see *Specify XML encryption policy (for SAML 2.0)* on page 624).

Steps

1. Select the primary XML decryption key from the list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see).

2. Optional: Select the secondary XML decryption key from the list.

Review credential settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review an IdP connection

About this task

When you finish creating or modifying a connection, you can review the connection settings and toggle the connection status on the **Activation & Summary** screen.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Save**.
- To discard your changes, click **Cancel**.

Results

> **Important:** When creating a new connection, the default connection status is **Enabled** when you reach the **Activation & Summary** screen.
>
> Regardless of whether you choose to disable a new connection now or later, you must click **Save** on the **Activation & Summary** screen if you want to keep the new connection.

The **SSO Application Endpoint** provides a sample URL at the `/sp/startSSO.ping` application endpoint that webmasters or web application developers at your site may use to invoke SSO for the connection. For a list of supported parameters, see *View SP application endpoints* on page 590.

If you have selected the **No Mapping** option on the **Identity Mapping** screen, the **Summary & Activation** screen does not show the **SSO Application Endpoint** sample URL.

OpenID Connect Relying Party support

PingFederate is capable of leveraging identities from OpenID Providers (OPs) to complete Browser 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. In essence, PingFederate retrieves identity information from the OP and passes the end-user claims, which are basically 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, a modern standard that has been gaining momentum in the industry.

If the OP supports the OpenID Connect Discovery specification, the connection setup can be 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.

In addition, the protocol settings can be refreshed 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 could be mapped to the target applications later. In the rare 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 may 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 may define additional request parameters, which could be included in the authentication requests to support OP-specific use cases. Administrators may restrict the values to those defined in the configuration. Alternatively, administrators may allow the target applications to optionally override the values at runtime. Furthermore, 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 via multiple query parameters, unsigned. Optionally, administrators can configure PingFederate to send request parameters via a request object by value, in which case request parameters are represented by individual claims in a signed 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` SP application endpoint or the `/sp/init_login.ping` SP protocol endpoint.
2. PingFederate (the RP) sends to the OP via the browser an authentication requests 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) via 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) via the browser.

   The access token (if any) may 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 that 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*.

Create an OpenID Connect IdP connection

About this task

Steps

1. If you have not already done so, enable the OpenID Connect protocol for IdP connections.
   a. Go to the **System**# **Protocol Settings**# **Roles & Protocols** screen.
   b. Ensure the SP role is activated.
   c. Select the **OpenID Connect** check box to activate the protocol for IdP connections.
2. On the **Service Provider** menu, create a new IdP connection.
3. On the **Connection Type** screen, select the **Browser SSO Profiles** check box and select **OpenID Connect** from the **Protocol** list.

   Note that when OpenID Connect is the chosen protocol, the other types become unavailable.
4. On the **Connection Options** screen, you may enable JIT provisioning, OAuth attribute mapping (which requires the OAuth 2.0 authorization server role), or both.

   For simplicity, this topic focuses only on managing OpenID Connect IdP connection settings.
5. On the **General Info** screen:
   a. Provide the required information, which includes

      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, please refer to 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 (see *step 12*).
   b. Optional: Click **Load Metadata**.

      Tip: Loading metadata from the OP expedites the connection setup. You may also update an existing connection by reloading metadata.
6. On the **Browser SSO** screen, click **Configure Browser SSO**.
7. On the **User-Session Creation** screen, click **Configure User-Session Creation**.

8. On the **Identity Mapping** screen, you have three choices:

   - 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.
9. On the **Attribute Contract** screen, 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** screen, the attribute contract is populated automatically.
10. On the **Target Session Mapping** screen, 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 have the options to query additional attributes from a data store, to specify issuance criteria, or to do both. When mapping an attribute, select **Provider Claims** from the **Source** list to map the attribute to an end-user claim.

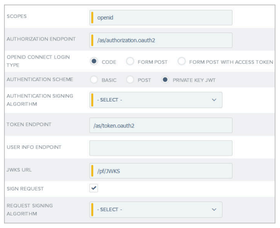
    If your target application requires the associated access token, select **Context** as the source and **Access Token** as the value.

    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 12*, such that the OP will return an access token and an ID token as part of the authentication and authorization flow.

    Note that the **Target Session Mapping** configuration does not apply when the **No Mapping** option is chosen on the **Identity Mapping** screen.
11. On the **Protocol Settings** screen, click **Configure Protocol Settings**.

12. On the **OpenID Provider Info** screen, provide the scopes, the endpoints, and the authentication scheme; for example:



SCOPES	openid
AUTHORIZATION ENDPOINT	/as/authorization.oauth2
OPENID CONNECT LOGIN TYPE	[✓ CODE] FORM POST FORM POST WITH ACCESS TOKEN
AUTHENTICATION SCHEME	BASIC POST [✓ PRIVATE KEY JWT]
AUTHENTICATION SIGNING ALGORITHM	- SELECT -
TOKEN ENDPOINT	/as/token.oauth2
USER INFO ENDPOINT	
JWKS URL	/pf/JWKS
SIGN REQUEST	✓
REQUEST SIGNING ALGORITHM	- SELECT -

Note: If you have chosen to load the metadata from the OP on the **General Info** screen, 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 authentication and 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 may enter a relative path (begin with a forward slash) if you have provided a base URL on the **General Info** screen. 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**.

Field	Description
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 via HTTP Basic authentication. ▪ Select **POST** to submit credentials via POST. ▪ Select **Private Key JWT** to authenticate via the private_key_jwt Client Authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication). The default selection (without loading metadata from the OP) is **Basic**.
Authentication Signing Algorithm	Select the algorithm that PingFederate uses to sign the JWT. Applicable and visible only when **Private Key JWT** is the chosen authentication scheme. If PingFederate is either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . **Note:** Based on the chosen signing algorithm, PingFederate selects the signing JSON Web Key (JWK) from its JWK Set (JWKS) at runtime. In order for the OP to validate the signed JWT, ensure that the OP can access your PingFederate JWKS endpoint, which returns the current set of JSON Web Keys. The PingFederate JWKS endpoint is located at `<Base URL>/pf/JWKS`, where **Base URL** is defined on the **System**# **Protocol Settings**# **Federation Info** screen. 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 OpenID Provider configuration endpoint to obtain the information (see).

Field	Description
Token Endpoint, UserInfo Endpoint, and JWKS URL	Various 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 in order 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 11 runtime environment or integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see .) 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. This check box is not selected by default, in which case PingFederate sends request parameters via 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 either deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration and static keys, see and , respectively.) ⓘ **Note:** If static keys for OAuth and OpenID Connect are enabled, EC algorithms that have not been configured with an active static keys are hidden. Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see . ⓘ **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 the **System# Protocol Settings# Federation Info** screen. 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 (see).

13. Optional: Remain on the **OpenID Provider Info** screen and specify the request parameters that are allowed to be included in the authentication requests to the OP under **Request Parameters** (see).
14. Optional: On the **Overrides** screen, specify a default target URL and authentication context overrides.
15. On the **Activation & Summary** screen, review your connection settings.

    When you finish setting up a connection, you may 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** screen for a new connection if you want to keep the configuration.

    You can deactivate a connection at any time (for maintenance, for example). When a connection is inactive, all transactions to or from this partner are disabled.

    In this use case, because PingFederate is essentially 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** screen. This registration should be associated with the client that represents PingFederate, the client that you have provided on the **General Info** screen. For more information, please refer to 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 may use to invoke SSO for the connection. For a list of supported parameters, see *View SP application endpoints* on page 590.

If you have selected the **No Mapping** option on the **Identity Mapping** screen, the **Summary & Activation** screen 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 *Configure request parameters and SSO URLs* on page 669.

Configure request parameters and SSO URLs

About this task

On the **OpenID Provider Info** screen, administrators may define request parameters under **Request Parameters** for the following purposes:

- Allow custom request parameters to be include in the authentication requests to support 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. (Note that the OP may 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).

Follow these steps to define one or more request parameters:

Steps

1. Add a request parameter under **Name**.
2. Define a default parameter value under **Value**.

   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` SP application endpoint. If the target application does not provide the parameter in its SSO URL (and no default value is specified), the parameter is not included in the authentication requests.

   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.

3. Select the check box under **Application Endpoint Override** if the target application is allowed to override the parameter value at runtime.

   If the target application does not provide the parameter in its SSO URL and the configuration does not include a default value, the parameter is not included in the authentication requests.

   If the target application does not provide the parameter in its SSO URL, the default value (if any) is used.

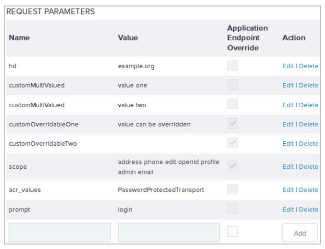
   If the target application provides the parameter in its SSO URL to the `/sp/startSSO.ping` SP application endpoint, the value in the SSO URL is used.

   Note that the `/sp/init_login.ping` SP protocol 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 for more information.
4. Click **Add**.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change. Use the **Delete** and **Undelete** workflow to remove an entry or cancel the removal request.
5. Repeat these steps to define another request parameters.

Example

Consider the following sample configuration:



REQUEST PARAMETERS

Name	Value	Application Endpoint Override	Action
hd	example.org		Edit | Delete
customMultiValued	value one		Edit | Delete
customMultiValued	value two		Edit | Delete
customOverridableOne	value can be overridden	✓	Edit | Delete
customOverridableTwo		✓	Edit | Delete
scope	address phone edit openid profile admin email	✓	Edit | Delete
acr_values	PasswordProtectedTransport		Edit | Delete
prompt	login		Edit | Delete
			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** screen; for example:

  https://sso.example.com/sp/startSSO.ping?PartnerIdpId=https%3A%2F%2Fsso.alpha.local%3A9031& **customOverridableOne=foo**

  To construct a multivalued request parameter, append the request parameter multiple times with different values; for example:

  https://sso.example.com/sp/startSSO.ping?PartnerIdpId=https%3A%2F%2Fsso.alpha.local%3A9031& **customOverridableOne=foo** *&* **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; therefore, 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 screen) 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 that while the target application can request different scopes, the OP may 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`.

Note: These examples use the `/sp/startSSO.ping` SP application endpoint. As needed, you may 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: When constructing SSO URLs, parameters with restricted characters must be URL-encoded.

For information about URL encoding, please refer to third party resources, such as *HTML URL-encoding Reference* (www.w3schools.com/tags/ref_urlencode.asp).

Query parameters versus request object

By default, PingFederate sends all request parameters via multiple query parameters, unsigned. If the **Sign Request** check box is selected, PingFederate creates a signed JWT that contains claims representing the request parameters and passes the signed JWT as one query parameter, request, to the OP. Note that 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. (Note that the client authenticates via the HTTP Basic authentication scheme and initiates 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 that the client_id, response_type, and scope request parameters are always passed to the OP as individual query parameter 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*.

Configure IdP discovery using a persistent cookie

About this task

PingFederate's proprietary IdP-discovery method makes use of an IdP persistent reference cookie (IPRC) to track the identity provider with whom a user last authenticated. There are three significant differences between standard IdP Discovery and the IPRC method:

- Standard IdP Discovery may be used only with SAML 2.0; the IPRC may be used with any federation protocol.

- The common domain cookie (CDC) may 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 readable by both federation partners; the IPRC is set by the SP, using information in the SAML assertion, and cannot be accessed by the IdP.

Note that the deployed connection configuration between SP and IdP partners must include SP-initiated 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 (default: `IdPId`). Note that the cookie name cannot contain any of the following characters: `&`, `>`, `<`, comma, semicolon, space.
IdpIdCookieLifeTimeInDays	The lifetime for the cookie (default: `365` days, maximum: `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:** Once 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.

WS-Trust STS configuration

The PingFederate WS-Trust Security Token Service (STS) provides security-token validation and creation to extend SSO access to identity-enabled web services (see *Web services standards* on page 47).

The section provides instructions for configuring the WS-Trust STS.

Server settings

To use the PingFederate WS-Trust STS for partner connections, start by enabling the WS-Trust protocol on the **System**# **Protocol Settings**# **Roles & Protocols** screen. Once the protocol is enabled, 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 corresponding browser-based SSO protocols).

In addition, also under Server Settings, you have the option of requiring authentication globally for access to STS endpoints.

Enable the WS-Trust protocol

About this task

Steps

1. Go to the **System**# **Protocol Settings**# **Roles & Protocols** screen.

2. If you have not done so, enable the applicable federation role (or roles) for your deployment.
3. Under the role (or roles) that you require STS processing, select the **WS-Trust** check box to enable the protocol.

   Note: PingFederate supports the STS with or without selections of other browser-based SSO protocols. The handling of SAML 1.1 and 2.0 tokens is independent of the supported browser-based SSO protocols shown on the same screen.

4. Enter your SAML federation IDs on the **System# Protocol Settings# Federation Info** screen (unless these IDs are already established for 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 as needed.

Configure STS authentication

About this task

On the **System# Protocol Settings# WS-Trust STS Settings** screen, you may configure PingFederate to require that client applications provide credentials to access the STS.

While this is an optional configuration, it is recommended for 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 may 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 SP partner connections, or token-to-token mappings, using more fine grained controls, at the connection level via Issuance Criteria.

Steps

1. On the **WS-Trust STS Settings** screen, click **Configure WS-Trust STS Authentication**.
2. On the **Authentication Methods** screen, select HTTP Basic, mutual SSL/TLS authentication, or both.

   If both HTTP Basic and mutual SSL/TLS authentication are selected, *all* clients must provide credentials for both mechanisms.

   Important: If you choose mutual SSL/TLS authentication, you must configure a secondary PingFederate HTTPS port (pf.secondary.https.port) in the `run.properties` file (see *Configuring PingFederate properties* on page 221).

3. If you have chosen HTTP Basic, manage user accounts on the **HTTP Basic Authentication** screen.

   Click **Create User** and enter a username and its password on the **User Account** screen. Repeat to create additional user accounts for your client applications.

   On the **HTTP Basic Authentication** screen, you can also delete user accounts and update their passwords.

4. If you have chosen mutual SSL/TLS authentication, click **Configure Mutual SSL Authentication** on the **Mutual SSL Authentication** screen.
   a. On the **Authentication Options** screen, select to restrict access by the subject DN or issuer of the client certificate.

      If both options are selected, the client certificate used for authentication to the STS endpoints must meet both sets of restrictions.
   b. If you have chosen to restrict by the subject DN, enter one or more subject DNs on the **Allowed Subject DNs** screen.

      On the **Allowed Subject DNs** screen, you may edit or delete existing entries but you must keep at least one subject DN.
   c. If you have chosen to restrict by the certificate issuer, select one or more client certificate on the **Allowed Issuer Certificates** screen.

      If you have not yet imported the client certificate, click **Manage Certificates** to do so.

      On the **Allowed Issuer Certificates** screen, you may remove existing entries but you must keep at least one issuer.
   d. On the **Summary** screen, review your mutual SSL/TLS authentication settings and then click **Done**.
5. When you finish configuring WS-Trust STS settings, you can review the configuration on its **Summary** screen.

   If you want to keep your changes, click **Save**.

Identity provider STS configuration

This section covers the IdP configuration for the PingFederate WS-Trust STS.

Manage token processors

About this task

The PingFederate Security Token Service (STS) uses token processors to validate incoming tokens and token requests. 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* on page 47.)

PingFederate comes bundled with the following token processors:

- JWT Token Processor
- Kerberos Token Processor
- OAuth Bear Token Processor
- SAML 1.1 Token Processor
- SAML 2.0 Token Processor
- Username Token Processor

You can also deploy additional token translators from *Ping Identity website* (www.pingidentity.com/en/products/downloads.html).

You manage token processor instances on the **Identity Provider# Token Processors** screen.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Results

ⓘ **Note:** Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.

If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

For simplicity, this topic focuses on configuring an instance of one of the integrated token processors. For add-on processors, please refer to the online *documentation* referenced in the download package.

Select a token processor type

About this task

The first step in creating a token-processor instance is choosing the processor type.

Steps

- On the **Type** screen, configure the basics of this token processor instance.
  a. Enter a name and ID for this token processor instance.
  b. Select the token-processor type from the list.
  c. Optional: 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 screens.

Configure a token processor instance

About this task

Depending on the selected token processor, the **Instance Configuration** screen presents you with different parameters.

Steps

- For token processors bundled with PingFederate, refer to one of the following sections:
  - *Configure a Username Token Processor instance* on page 676
  - *Configure a Kerberos Token Processor instance* on page 677
  - *Configure an OAuth Token Processor instance* on page 677
  - *Configure a JSON Web Token Processor instance* on page 678
  - *Configure a SAML Token Processor instance* on page 678
- For add-on processors, please refer to the online *documentation* referenced in the download package.

Configure a Username Token Processor instance

About this task

The integrated Username Token Processor accepts and validates username security tokens.

Steps

- On the **Instance Configuration** screen, configure the basics of this token processor instance.
  a. If you have not yet defined the desired Password Credential Validator instance, click **Manage Password Credential Validators** to do so.
  b. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.
  c. Select a Password Credential Validator instance from the list and 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 is able to 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 lockout those accounts in their source locations.
  d. Enter a value in the **Authentication Attempts** field.

     When the number of login failures reaches this threshold, the user is locked out for a period time.

     The default value is `3`.

Configure a Kerberos Token Processor instance

About this task

The integrated Kerberos Token Processor accepts and validates Kerberos tokens via a configured Kerberos realm. Moreover, it supports authentication mechanism assurance from Active Directory domain service, thus making it possible to restrict access to users authenticating through specific mechanisms. For more information, see *Authentication mechanism assurance* on page 730.

Steps

- On the **Instance Configuration** screen, select the applicable domain from the **Domain/Realm Name** list.

  An Active Directory 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 *Configuring Active Directory domains or Kerberos realms* on page 209.)

  Note: Kerberos tickets can be accepted from domains other than the domain configured in the Token Processor, provided that there is a transient, two-way trust. This trust exists by default when domains are joined within a single server forest (see *Multiple-domain support* on page 210).

Configure an OAuth Token Processor instance

About this task

The PingFederate STS provides validation for OAuth 2.0 bearer tokens. Generally, a client would send a base64-encoded access token in order to receive a SAML token in exchange. To use this token processor, you must first configure an Access Token Management (ATM) instance.

(For more information about PingFederate OAuth authorization server and access token management, see *About OAuth* on page 103 and *Access token management* on page 445.)

Steps

- On the **Instance Configuration** screen, configure the basics of this token processor instance.
  a. Select an ATM instance from the list.

     If the desired ATM instance is not shown, click **Manage Access Token Manager**.

     The token processor instance uses the selected ATM instance to validate the OAuth bearer access tokens.
  b. Optional: Select the **Scope Value as Single String** check box.

     If selected, the scope value is returned as a single space-delimited set of string values; otherwise, scope values are returned as a multivalued attribute.

Configure a JSON Web Token Processor instance

About this task

The PingFederate STS provides validation for JSON web tokens (JWTs).

Steps

- On the **Instance Configuration** screen, provide the required information.

  Refer to the following table for information about each field.

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.

Configure a SAML Token Processor instance

About this task

The integrated SAML (1.1 or 2.0) Token Processor accepts and validates SAML (1.1 or 2.0) security tokens. The PingFederate STS validates digital signatures using all trusted certificate authorities (CAs) imported into PingFederate. As needed, you may restrict the signature verification process by subject DNs or issuers (or both) to limit the token requests accepted for this token processor instance.

In addition, you must indicate a unique identifier for the PingFederate STS. Once defined, incoming SAML tokens must contain this ID in its audience element in order for them to be accepted by this token processor instance.

Steps

- On the **Instance Configuration** screen, configure the basics of this token processor instance.
  a. Enter the URI that uniquely identifies your federation gateway for this SAML protocol in the **Audience** field.

     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.
  b. Optional: Click **Add a new row to 'Valid Certificate Issuer DNs'** and then enter one or more issuers.

     If issuer DNs are specified here, then only those issuers are considered valid for verifying incoming digital signatures. Otherwise, all trusted certificate authorities (CAs) are used to verify signatures.
  c. Optional: Click **Add a new row to 'Valid Certificate Subject DNs'** and then enter one or more subject DNs.

     If subject DNs are specified here, then only those subject DNs are considered valid for verifying incoming digital signatures. Otherwise, all trusted certificate authorities (CAs) are used to verify signatures.

  > **Important:** 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.

Extend a token processor contract

About this task

Token processors allow administrators to add to a built-in list of user attributes that the processor returns from an incoming token; it is essentially an extended processor attribute contract. Note that the **Extended Contract** screen shows a different list of attributes under **Core Contract**, depending on the token processor selected.

Steps

- Enter the name of the desired attribute and 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 this step as needed to add another attribute.

Set attribute masking

About this task

On the **Token Attributes** screen, you can choose to mask attribute values that PingFederate logs from this processor instance at runtime.

Steps

- Optional: Under **Mask Log Values**, select the attribute whose value you want to mask in logs.

  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 related check box under the list of attributes.

Review the token processor configuration

About this task

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Manage STS request parameters

About this task

As an option for configuring PingFederate to act as a WS-Trust STS, you can define sets of RST metadata parameters that can be used to map attribute values into issued security tokens. After these *request contracts* are defined, you can make them available when configuring WS-Trust STS settings in the SP connections (see *Select a request contract* on page 683).

You manage request contract on the **Identity Provider#** **STS Request Parameters** screen.

Steps

- To configure a new set of request parameters, click **Add New Request Contract**.
- To modify an existing request contract, select it by its name under **Contract Name**.
- To review the usage of an existing request contract, click **Check Usage** under **Action**.
- To remove an existing request contract or to cancel the removal request, click **Delete** or **Undelete** under **Action**.

Create a request contract

About this task

On the **Create Request Contract** screen, identify the contract and define parameters that will be available in token requests (as associated with this contract for partner connections).

Steps

1. Enter the contract name and contract ID.

   Once the request is saved, the name and ID cannot be modified.
2. Enter the parameter that will be included in RSTs and click **Add**.

   You must add at least one parameter. Repeat this step to add more parameters.

   Once the request is saved, you may add, modify, or remove parameters but you must keep at least one parameter.

Configure SP connections for STS

About this task

You can configure an STS connection to an SP partner either in conjunction with browser-based SSO or independently.

Steps

- To configure an STS connection, select the **WS-Trust STS** check box.

  The **WS-Trust STS** option is only available after you enable the **WS-Trust** role on the **System# Protocol Settings# Roles & Protocols** screen.

Configure protocol settings for IdP STS

About this task

Steps

1. Enter a URL for your partner's web service in the text field and then click **Add**.

   This identifier is compared to the AppliesTo element in the Requests for Security Token (RST) messages and may be either a complete URL or a base URL for matching variable ports or paths.

   Repeat this step to add additional identifiers as needed.
2. Select any of the following 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* on page 102 for more information on the use of these token-type requests.
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* (docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SAMLTokenProfile-v1.1.1-os.html). 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.

Set a token lifetime

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

- Optional: Override the default values for the following fields:

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.

Configure token creation

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 (the "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 SSO, the mapping can be augmented using local data stores, variable or constant text, or expressions.

Details of this configuration are handled under the **Token Creation** task.

Steps

- To continue, click **Configure Token Creation**.

Define an attribute contract for IdP STS

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 (see). You identify these attributes on the **Attribute Contract** screen.

Steps

1. Enter the attribute name in the text box.

Attribute names are case-sensitive and must correspond to the attribute names (including claims) expected by the requesting WSC.

> **Tip:** The Format attribute associated with the NameID element in outgoing SAML tokens may be set when needed by adding an attribute called SAML_NAME_FORMAT. The value of that attribute can then be mapped later (see *Configure contract fulfillment for token creation* on page 686).
>
> 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 SP (if required) the type of credentials used to authenticate to the IdP application—authentication context. Map a value for

the authentication context on the attribute-mapping screen later in the configuration, from any available attribute source, including the RST if a requested context is specified as a request parameter (see *Configure contract fulfillment for token creation* on page 686).

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** on the **WS-Trust STS# Protocol Settings** screen.

   Change the default namespace selection if you and your SP partner have agreed to a specific namespace.

   > **Note:** As needed, 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. (Restart of PingFederate is required to activate any changes made to this file.)

   (For more information about attribute namespace, see *Attribute contracts* on page 121.)
3. Click **Add**.
4. Repeat until all applicable attributes are defined.

Results

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.

Select a request contract

About this task

This optional setting on the **Request Contract** screen allows you to use XML parameters contained in RSTs for token-attribute mapping (see *Manage STS request parameters* on page 680).

Steps

- If you are not using request parameters, click **Next** to continue.
- To use request parameters, select the check box and choose a request contract from list.

  If the contract you want is not shown, click **Manage STS Request Parameters**.

  When selected, you may choose the request contract as the attribute source in the **IdP Token Processor Mapping** configuration later in the setup.

  If you are editing an existing configuration, you may change the request contract or disable this optional setting. These changes may require additional configuration changes in subsequent tasks.

Manage IdP token processor mappings

About this task

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.

You can map one or more token processor instances into an SP connection to satisfy multiple session-management requirements where needed. (The same token processor instances may also be mapped in multiple SP connections.)

When token processor instances are restricted to certain virtual server IDs, the allowed IDs are displayed under **Virtual Server IDs**.

Steps

- To map a token processor instance, click **Map New Token Processor Instance**.
- 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**.
- 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**.

Select a token processor instance

About this task

On the **Token Processor Instance** screen, choose an instance of a deployed token processor that suits your requirements for this connection.

Steps

1. Select a token processor instance from the list.

   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.

   When selected, the administrative console adds a new set of sub tasks (the **Override Instance** screen and its sub tasks).

   > **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 plug-in configurations* on page 119.

Results

If you are editing a currently mapped token processor 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.

Override a token processor instance

About this task

On the **Override Instance** screen, you start a series of sub tasks to 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

- Click **Override Instance Settings**.

  On each of the settings screens, select the **Override** check box, make your changes, and then click **Next**. When you are finished, click **Done** to continue with the rest of the mapping configuration.

> **Note:** The override setting screens are functionally identical to those used for creating a new token processor instance (see *Manage token processors* on page 675).

Results

If you are editing a currently mapped token processor instance, click **Override Instance Settings** to reconfigure any overridden settings for this connection. You may also remove all overridden settings on a per-screen basis by clearing the **Override** check box near the top of the screen.

Restrict a token processor to certain virtual server IDs

About this task

When you multiplex one connection for multiple environments (see), you can enforce authentication requirements by restricting a token processor to certain virtual server IDs on the **Virtual Server IDs** screen. By default, no restriction is imposed.

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 token processor.

Results

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.

Select an attribute retrieval method for token creation

About this task

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.

The values supplied by the token processor are shown under **Token Processor Contract** on the **Attribute Retrieval** screen.

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.

Steps

- Select the **Retrieve additional attributes from data stores ...** 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 ...** option if you do *not* require data store query.

Results

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.

Configure attribute sources and user lookup for token creation

About this task

Attribute sources are specific data store or directory locations containing information that may be needed for the attribute contract. 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.

Steps

- Click **Add Attribute Source** and then follow a series of sub tasks to complete the configuration.

  For step-by-step instructions, see *Choose a data store* on page 872.

  Repeat to add additional attribute sources.

Results

If you are editing a currently mapped token processor instance, you can add, remove, or reorder attribute sources, which may require additional configuration changes in subsequent tasks.

Configure contract fulfillment for token creation

About this task

On the **Attribute Contract Fulfillment** screen, map values to the attributes defined for the contract. These are the values that will be included in the SAML security tokens sent to the SP.

Steps

- For each attribute, select a source from the list and then choose or enter a value.
  - **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 (see **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# Protocol Settings# WS-Trust STS Settings** screen.
  - **Request**

    When selected, the **Value** list is populated 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** screen. 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 is populated 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 SAML security token.
- **Expression** (when enabled)

  This option provides more complex mapping capabilities; for example, 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 (see **Text**).

  Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
- **Text**

  When selected, the text you enter is mapped to the value of the attribute in the SSO 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 this syntax:

  `${ds.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.

All attributes must be mapped.

Results

If you are editing a currently mapped token processor instance, you can update the mapping configuration, which may require additional configuration changes in subsequent tasks.

Define issuance criteria for token creation

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that 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 (see), 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 (see).

ⓘ **Note:** Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.

   ⓘ **Note:** If you want 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 the **System**# **Protocol Settings**# **WS-Trust STS Settings** screen.

3. Select the comparison method under **Condition**.

   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 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.

   Note: If you want 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 (for example, case sensitivity, or whitespace).

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 *Simple Object Access Protocol* (www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383507).

   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.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the IdP token processor mapping

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Select a request error handling method

About this task

If you are using request parameters to fulfill the attribute contract and the parameter values are not supplied in hte RST messages, you can choose whether to continue or abort the token-creation process.

Note that the **Error Handling** screen is presented only if a request contract is configured on the **Request Contract** screen for this connection.

Review the token creation configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the IdP STS settings

Steps

- To amend your configuration, click the corresponding screen title and 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 may 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 SP configuration for STS.

Manage token generators

About this task

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 in order to set up an STS connection or a token-to-token mapping.

(For more information about WS-Trust, see *Web services standards* on page 47.)

PingFederate comes bundled with the SAML 1.1 Token Generator and SAML 2.0 Token Generator.

You can also deploy additional token translators from *Ping Identity website* (www.pingidentity.com/en/products/downloads.html).

You manage token generator instances on the **Service Provider**# **Token Generators** screen.

Steps

- 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 retain any configuration changes, click **Save**.
- To discard any configuration changes, click **Cancel**.

Results

> ⓘ **Note:** Automatic multi-connection error checking occurs by default whenever you access this screen. The intent is to verify that configured connections have not been adversely affected by changes made here.
>
> If you experience noticeable delays in accessing this screen, you can optionally disable automatic connection validation on the **Identity Provider**# **SP Connections** screen or the **Service Provider**# **IdP Connections** screen.

For simplicity, this topic focuses on configuring an instance of the SAML 1.1 or 2.0 Token Generator. For add-on token generators, please refer to the online *documentation* referenced in the download package.

Select a token generator type

About this task

The first step in creating a token-generator instance is choosing the generator type.

Steps

- On the **Type** screen, configure the basics of this token generator instance.
  a. Enter a name and ID for this token generator instance.
  b. Select the token-generator type from the list.
  c. Optional: 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 screens.

Configure a token generator instance

About this task

Depending on the selected token generator, the **Instance Configuration** screen presents you with different parameters.

Steps

- For the integrated SAML 1.0 and 2.0 Token Generators, refer to the following table and specify parameters for generated SAML tokens.

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 on the **System**# **Protocol Settings**# **Roles & Protocols** screen.
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** (see *Manage digital signing certificates and decryption keys* on page 299).
Signing Algorithm	Select the signing algorithm corresponding to the selected certificate. Choices include SHA1 for both RSA and DSA, RSA-SHA256, SHA384, and SHA512; as well as ECDSA-SHA256, SHA384, and SHA512.
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 instead.
Include Raw Key in KeyValue	If selected, the raw key is included in the KeyInfo element as well.
Audience	This is a unique identifier for the target web service, used for the audience element of the generated SAML token.
Confirmation Method	Choose from among available methods: ▪ **urn...cm:sender-vouches** (default) ▪ **urn...cm:bearer** ▪ **urn...cm:holder-of-key** For more information, see *WSS SAML Token Profile* (docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SAMLTokenProfile-v1.1.1-os.html).

Field	Instructions
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 (see *Manage certificates from partners* on page 310).

- For add-on generators, please refer to the online *documentation* referenced in the download package.

Extend a token generator contract

About this task

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.

Token generators allow administrators to add to a built-in list of user attributes that the generator includes in the outgoing token; it is essentially an extended generator-attribute contract. Note that the **Extended Contract** screen shows a different list of attributes under **Core Contract**, depending on the token generator selected.

Steps

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Review the token generator configuration

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Configure IdP connections for STS

About this task

You can configure an STS connection to an IdP partner either in conjunction with browser-based SSO or independently.

Steps

- To configure an STS connection, select the **WS-Trust STS** check box.

  The **WS-Trust STS** option is only available after you enable the **WS-Trust** role on the **System# Protocol Settings# Roles & Protocols** screen.

Configure protocol settings for SP STS

About this task

Steps

- On the **Protocol Settings** screen, choose whether to only validate incoming SAML tokens or to validate and then also generate local tokens to enable SSO access to web services at your site.

  If you choose to generate local tokens as well, you must set up at least one token generator.

Configure token generation

About this task

For the PingFederate STS to issue a security token that meets identity requirements of web services at your site, you must indicate what user attributes are included in the incoming token. As with Browser SSO, the mapping can be augmented using local data stores, variable or constant text, or expressions.

Details of this configuration are handled under the **Token Generation** task.

Steps

- To continue, click **Configure Token Creation**.

Define an attribute contract for SP STS

About this task

An attribute contract is the set of user attributes expected in incoming SAML assertions (see *Attribute contracts* on page 121). You identify these attributes on the **Attribute Contract** screen.

Optionally, you can mask the values of attributes (other than SAML_SUBJECT) in logs that PingFederate writes when it receives security tokens.

Steps

1. Enter the attribute name in the text box.

   Attribute names are case-sensitive and must correspond to the attribute names expected by the requester.
2. Optional: Select the check box under **Mask Values in Log**.
3. Click **Add**.
4. Repeat until all applicable attributes are defined.

Results

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.

Manage SP token generator mappings

About this task

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.

As needed, you can map one or more token generator instances into an IdP connection to satisfy different token requirements by the web services at your site. (The same token generator instances may also 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

- To map a token generator instance, click **Map New Token Generator Instance**.
- To edit the mapping configuration of a token generator instance, open it by clicking on its name, select the setting that you want to reconfigure, and complete the change.
- To remove a token generator instance 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 configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Select a token generator instance

About this task

On the **Token Generator Instance** screen, choose an instance of a deployed token processor that suits your requirements for this connection.

Steps

1. Select a token generator instance from the list.

   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.
2. Select the **Override Instance Settings** check box if you want to customize one or more token processor settings for this connection alone.

   When selected, the administrative console adds a new set of sub tasks (the **Override Instance** screen and its sub tasks).

   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 plug-in configurations* on page 119.

Results

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.

Override a token generator instance

About this task

On the **Override Instance** screen, you start a series of sub tasks to override token generator settings specifically for this 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

- Click **Override Instance Settings**.

  On each of the settings screens, select the **Override** check box, make your changes, and then click **Next**. When you are finished, click **Done** to continue with the rest of the mapping configuration.

> ⓘ **Note:** The override setting screens are functionally identical to those used for creating a new token generator instance (see *Manage token generators* on page 691).

Restrict a token generator to certain virtual server IDs

About this task

When you multiplex one connection for multiple environments (see), you can enforce integration requirements by restricting a token generator to certain virtual server IDs on the **Virtual Server IDs** screen. By default, no restriction is imposed.

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 token generator.

Results

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.

Select an attribute retrieval method for token generation

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 under **Attribute Contract** on the **Attribute Retrieval** screen.

Steps

- 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 *Choose a data store* on page 872.

Results

If you are editing a currently mapped token generator instance, you can change the mapping method, which may require additional configuration changes in subsequent tasks.

Configure contract fulfillment for token generation

About this task

On the **Token Generator Contract Fulfillment** screen, map values to the attributes defined for the contract. These are the values that the web services require.

Steps

- For each attribute, select a source from the list and then choose or enter a value.
  - **Assertion**

    When selected, the **Value** list is populated 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 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 local 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 (see **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**# **Protocol Settings**# **WS-Trust STS Settings** screen.
  - **LDAP**, **JDBC**, or **Other**

    When selected, the **Value** list is populated 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; for example, 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 (see **Text**).

    Note that expressions are not enabled by default. For more information about enabling and editing OGNL expressions, see .
  - **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 data store, when applicable, using this syntax:

    `${ds.`*`attribute`*`}`

    where *`attribute`* is any of the attributes that you have selected from the data store.

  All attributes must be mapped.

Results

If you are editing a currently mapped token generator instance, you can update the mapping configuration, which may require additional configuration changes in subsequent tasks.

Define issuance criteria for token generation

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that 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 (see), 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 (see).

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.

   > ⓘ **Note:** If you want 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 the **System**# **Protocol Settings**# **WS-Trust STS Settings** screen.

3. Select the comparison method under **Condition**.

   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 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.

   > ⓘ **Note:** If you want 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 (for example, case sensitivity, or whitespace).

4. Enter the desired (compared-to) value under **Value**.

   > ⓘ **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 *Simple Object Access Protocol* (www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383507).

   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.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.

8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

   > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the SP token generator mapping

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the token generation configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.

- To discard your changes, click **Cancel**.

Review the SP STS configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may 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

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 via an SP adapter on the same PingFederate server. Generally, these cases involve SaaS providers who may not support standards-based SSO but do provide proprietary SSO with “delegated authentication” (for example, 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 (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. Note that you may reuse instances that are also in use for connection configurations.

Manage mappings

About this task

On the **Adapter-to-Adapter Mappings** screen you can add, modify, or delete adapter-to-adapter mappings. The **Adapter-to-Adapter Mappings** screen is located under both the **Identity Provider** and **Service Provider** menu.

Steps

- To add a mapping, select an IdP adapter instance from the **Source Instance** list, an 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 vice versa.
- 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.

Assign a license group

About this task

Adapter-to-adapter mapping is considered a connection for licensing purposes. If your PingFederate license manages connections by groups, select a license group for this mapping configuration.

Note: This screen is not displayed for unrestricted or other types of licenses.

Steps

- Select the license group from the list.

Configure attribute lookup for adapter-to-adapter mapping

About this task

Attribute sources are specific data store or directory locations containing information that may be required to fulfill the SP adapter contract. This optional adapter-to-adapter configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

Steps

- To configure an attribute source, click **Add Attribute Source** and complete the setup steps (see *Choose a data store* on page 872).
- 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 may need to modify dependent data in subsequent steps, as indicated.

Configure contract fulfillment for adapter-to-adapter mapping

About this task

The next step in this configuration is to map values from the IdP adapter into the attributes required by the SP adapter (the Adapter Contract).

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 screen* on page 864). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 859. For syntax and examples, see sections under *Construct OGNL expressions* on page 860.)
- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 859). All of the variables available for text entries (see below) are also available for expressions.
- 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.

  You can also enter values from your data store, when applicable, using this syntax:

  `${ds.`*`attr-source-id.attribute`*`}`

  where *`attr-source-id`* is the **Attribute Source ID** value (see *Choose a data store* on page 872) and attribute is any of the data store 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.

Configure a default target URL (optional)

About this task

Use this screen to assign a default target URL for this adapter-to-adapter mapping. Entering a URL in the Default Target URL field overrides any SP Default URL SSO setting (see *Configure default URLs* on page 589).

Note: If specified, the adapter-to-adapter endpoint parameter TargetResource overrides any default target URL for an adapter-to-adapter mapping (see *System-services endpoints* on page 773).

Define issuance criteria for adapter-to-adapter mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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.

Source	**Description**
JDBC, LDAP, or other types of data store (if configured)	Select to evaluate attributes returned from a data source.
Mapped Attributes	Select to evaluate the mapped attributes.

2. Select the attribute to be evaluated under **Attribute Name**.
3. Select the comparison method under **Condition**.

   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 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.

4. Enter the desired (compared-to) value under **Value**.

   ⓘ **Note:** Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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; for example, display an error message or e-mail an administrator.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.

6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
    a. Click **Show Advanced Criteria**.
    b. Enter the required expressions in the **Expression** field.
    c. Optional: Enter an error code or an error message in the **Error Result** field.

    > **Note:** If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.

    d. Click **Add**.
    e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
    f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the 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** screen.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Token translator mappings

This configuration is provided for use cases in which the PingFederate WS-Trust STS exchanges one type of security token for another without requiring a SAML token to be generated in between (see *About WS-Trust STS* on page 100). Use this configuration, for example, to convert a user's Kerberos token to a third-party proprietary 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 *Enable the WS-Trust protocol* on page 673). Also, be sure to configure the required token-translator instances. Note that you may reuse instances that are also in use for STS connection configurations.

Manage token mappings

About this task

On the **Token Translator Mappings** screen you can add, modify, or delete token-to-token mappings. The **Token Translator Mappings** screen is located under both the **Identity Provider** and **Service Provider** menu.

Steps

- To add a mapping, select an token processor instance from the **Source Instance** list, an token geenrator 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 multiple IdP token processors and/or SP token generators are configured, you must provide the TokenProcessorId and/or the TokenGeneratorId query parameter(s) in the request (see *System-services endpoints* on page 773).

- 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.

Configure attribute lookup for token mapping

About this task

Attribute sources are specific data store or directory locations containing information that may be required to fulfill the contract of the token generator. This optional adapter-to-adapter configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

Steps

- To configure an attribute source, click **Add Attribute Source** and complete the setup steps (see *Choose a data store* on page 872).
- 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 may need to modify dependent data in subsequent steps, as indicated.

Configure contract fulfillment for token exchange mapping

About this task

The next step in this configuration is to map values from the token processor into the attributes required by the token generator (the Token Generator Contract).

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.
- 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 screen* on page 864). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 859. For syntax and examples, see sections under *Construct OGNL expressions* on page 860.)

- LDAP/JDBC/Other (when a data store is used)

  Values are returned from your data store (if used). When you make this selection, the Value list is populated by the attributes from the data store.
- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 859). All of the variables available for text entries (see below) are also available for expressions.
- 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 data store, when applicable, using this syntax:

  `${ds.attr-source-id.attribute}`

  where `attr-source-id` is the **Attribute Source ID** value (see *Fulfillment by data store queries* on page 870) and `attribute` is any of the data store attributes you select.

Steps

1. Select a source of the attribute under **Source** for each attribute in the contract of the token generator.
2. Specify an attribute under **Value** for each attribute.

Define issuance criteria for token translator mapping

About this task

On the **Issuance Criteria** screen, define the criteria that must be satisfied in order for PingFederate to process a request further. In essence, this *token authorization* feature provides the capability to conditionally approve or reject requests based on individual attributes.

You 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**, have dependency on the type of configuration; irrelevant sources are automatically hidden for your convenience. Once a source a selected, 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.

You can define multiple criteria, in which case *all* criteria must be satisfied in order 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 may contain multiple values. Such criterion is considered satisfied if *one* of the multiple values matches (or does not match) the specified value. It is worth noting that values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

Note: Regardless of whether the criteria are defined using the user interface, attribute mapping expressions, or both, *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. Select the source of the attribute under **Source**.

   Once selected, the **Attribute Name** list is populated with the associated attributes or properties. 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 data store (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.

2. Select the attribute to be evaluated under **Attribute Name**.

   Note: If you want 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 the **System**# **Protocol Settings**# **WS-Trust STS Settings** screen.

3. Select the comparison method under **Condition**.

   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 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.

   Note: If you want 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 (for example, case sensitivity, or whitespace).

4. Enter the desired (compared-to) value under **Value**.

   Note: Values are compared verbatim. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions.

5. Enter a custom error message under **Error Result**.

   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 *Simple Object Access Protocol* (www.w3.org/TR/2000/NOTE-SOAP-20000508/#_Toc478383507).

   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.

   If localized descriptions are required, enter a unique alias in the **Error Result** field (for example, `someIssuanceCriterionFailed`); 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.

   If it not defined, PingFederate returns `ACCESS_DENIED` when the criterion fails at runtime.
6. Click **Add**.
7. Optional: Repeat to add multiple criteria using the user interface.
8. If you require complex evaluations, including conditional criteria or partial matching, define them using attribute mapping expressions. (Attribute mapping expressions must be enabled.)
   a. Click **Show Advanced Criteria**.
   b. Enter the required expressions in the **Expression** field.
   c. Optional: Enter an error code or an error message in the **Error Result** field.

      Note: If the expressions resolve to a string value (rather than `true` or `false`), the returned value overrides the **Error Result** field value.
   d. Click **Add**.
   e. Optional: Click **Test**, enter values in the applicable fields, and verify the result meets your expectation.
   f. Optional: Repeat to add multiple criteria using attribute mapping expressions.

Review the token exchange mapping

About this task

When you have finished configuring a token exchange mapping, you can review the configuration on the **Summary** screen.

Steps

- To amend your configuration, click the corresponding screen title and then follow the configuration wizard to complete the task.
- To keep your changes, click **Done** and then click **Save** on the next screen.
- To discard your changes, click **Cancel**.

Bundled adapters

PingFederate comes bundled with a set of adapters:

- Identifier First Adapter
- HTML Form Adapter
- Kerberos Adapter
- OpenToken Adapter
- Composite Adapter
- HTTP Basic Adapter

Refer to subsequent topics for configuration steps.

Identifier First Adapter

When a variety of user types are authenticating 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.

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. (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 data store 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 enter a new one. If the user opts to enter a new identifier, the adapter adds that identifier to the cookie once 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.

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

As needed, the authentication context can be overridden by 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.)

Configure an Identifier First Adapter instance

About this task

Steps

1. Click **Identity Provider# Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.

3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.
4. On the **IdP Adapter** screen, configure your Identifier First Adapter instance.

   For more information about each field, refer to the following table.

Field	
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.

5. On the **Extended Contract** screen, 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 data store queries to fulfill the domain attribute (see *step 7*).

6. On the **Adapter Attributes** screen, configure the pseudonym and masking options.

> ⓘ **Note:** The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

a. 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 SP partners use pseudonyms for account linking.

   > ⓘ **Note:** A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.

b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.

c. 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** screen, configure the adapter contract for this instance with the following optional workflows:

- Configure one or more data sources for data store queries.
- Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.

9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

Identifier First Adapter and authentication policies

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 data store 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. Essentially, policy paths are created by having rules matching expected values of the domain attribute (or other extended attributes). 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, refer to the subsequent sample use case.

Configure a policy for multiple user populations

About this task

You can 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. Consider the following sample use case.

You are tasked to enforce different sets of authentication requirements for two sets of users, employees and external consultants.

Employees are given username@example.com email addresses; for example, asmith@example.com. User records are stored in a local directory server. Employees sign on through an HTML Form Adapter instance.

On the other hand, 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 ensure that the Identifier First Adapter instance can handle the scenario where users may enter their email address or just their username when setting up the Identifier First Adapter instance. Additionally, when accessing protected resources, your organization have agreed to send the user's email address in the security token.

You have already created the following components:

- An LDAP data store 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 via the LDAP data store. 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; for 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:

1. An expression-enabled PingFederate environment (*step 1*).
2. An authentication policy contract to carry the email address from your organization to your partners (*step 2*).
3. A JDBC data store connecting to the database that hosts username, email, and domain information (*step 3*).
4. An Identifier First Adapter instance with an attribute source lookup configuration and a contract fulfillment via expressions for the domain adapter attribute (*step 4*).
5. An authentication policy to route user requests to different authentication sources based on user populations (*step 5*).

To fulfill the requirements:

Steps

1. Enable expressions in PingFederate.

   For configuration steps, see *Enable and disable expressions* on page 859.
2. On the **Identity Provider#** **Policy Contracts** screen, create an authentication policy contract without any additional attributes.

3. On the **System# Data Stores** screen, create a JDBC data store connection to the database that hosts username and domain information.
4. Create an instance of the Identifier First Adapter instance.
   a. Follow steps 1 through 6 in *Configure an Identifier First Adapter instance* on page 710.

      Suppose you name the adapter instance **ID 1st**.
   b. On the **Adapter Contract Mapping** screen, click **Configure Adapter Contract**.
   c. On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source**.

      Let the administrative console guide you to complete the **Attribute Sources & User Lookup** configuration.

      1. On the **Data Store** screen, enter an ID and a name for the attribute source; for example, `domainInfo` and `Domain Info`, respectively. Then select the JDBC data store created in *step 3*.
      2. On the **Database Table and Columns** screen, select the applicable schema and table. Then select the dsDomain column and click **Add Attribute**.
      3. On the **Database Filter** screen, specify a filter to search by identifier that can handle identifiers in the format of an email address or a username; for example:

```
dsUid='${subject}' OR dsMail='${subject}'
```

      For more information about each step, see *Data store query configuration* on page 872.
   d. On the **Adapter Contract Fulfillment** screen, 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.
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 data store. In order 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* on page 860.
   e. On the **Issuance Criteria** screen, click **Next**.

      Depending on the actual use cases, you may add one or more issuance criteria.
   f. On the **Summary** screen, review and save your adapter instance configuration.

5. Create an authentication policy with rules to form policy paths based on results from domain attribute values returned by the Identifier First Adapter.

a. On the **Identity Provider# Policies** screen, click **Add Policy**.
b. On the **Policy** screen, enter a name (and optionally a description) for the policy.
c. Select the Identifier First Adapter instance created in *step 4*.
d. Click **Rules** to open the **Rules** dialog.
e. Add three rules as follows.

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

In general, add one rule for each expected domain attribute value.

f. 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* on page 345.

g. Click **Done** to close the **Rules** dialog.

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.

h. 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. Select **Adapter (ID 1st)** under **Source**.
2. Select **subject** under **Attribute**.
3. Click **Done** to close the **Incoming User ID** dialog.

For more information, see *Specifying an incoming user ID* on page 344.

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

i. Configure contract fulfillment for each authentication policy contract as follows.

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* on page 350.

j. Click **Done** and then **Save**.

Results

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.

HTML Form Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or an IdM 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 *Password Credential Validator* (PCV); for example, an LDAP Username PCV. This authentication mechanism validates credentials against a user repository via an instance of a PCV. Multiple PCV instances may be added to an instance of the HTML Form 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 HTML Form Adapter instance, PingFederate invokes the adapter if it does not find a valid *authentication session*. If the HTML Form Adapter does not finds a valid adapter session, it displays a sign-on page and prompts the user for credentials.

If *customer IAM* is configured and enabled, users can optionally register local accounts or sign on using third-party identity providers. If a user choose to sign on using local accounts, the credentials are validated using the designated Password Credential Validator instance (or instances). If validated, PingFederate generates the requested 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 allows you to use different customizable and localizable template files, define a logout path or a logout redirect page, notify users with password expiry information, allow users to change or reset their network passwords or redirect users to a company-hosted password management system, and enable self-service password reset, account unlock, and username recovery. All capabilities can be configured on a per-adapter instance basis.

PingFederate also tracks login attempts per adapter instance. This capability adds a layer of protection against brute force and dictionary attacks. When the **Challenge Retries** threshold is reached, the user is locked out for a period of time. The default value for the **Challenge Retries** setting is three. 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 five on the target directory server and the **Challenge Retries** setting is also set to five (or a higher value), the fifth sign-on attempt could potentially lock the user accounts on the directory server. The lockout period is controlled by the *Account Locking Service* on page 911.

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

As needed, the authentication context can be overridden by 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.)

Configure an HTML Form Adapter instance

About this task

Steps

1. Click **Identity Provider# Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.
4. On the **IdP Adapter** screen, configure your HTML Form Adapter instance as follows:
   a. If you have not yet defined the desired Password Credential Validator instance, click **Manage Password Credential Validators** to do so.
   b. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.
   c. Select a Password Credential Validator instance from the list and 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 is able to 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 lockout those accounts in their source locations.
   d. 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:** If you intend to enable PingFederate authentication sessions globally or individually for this adapter instance, select **None**. For more information about PingFederate authentication sessions, see *Sessions* on page 372 and *Configure authentication sessions* on page 375. 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 may configure the **Idle Timeout** setting for the same purpose (see *Configure authentication sessions* on page 375). 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 may configure the **Max Timeout** setting for the same purpose (see *Configure authentication sessions* on page 375). 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 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. For more information, see *Extend the lifetime of the PF cookie* on page 287. Alternatively, you can enable PingFederate authentication sessions, store the authentication sessions externally, and leverage them as users request protected resources after restarting their browsers. For more information, see *Sessions* on page 372.

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 may also provide your users the Change Password endpoint shown on the **Summary** screen. The Change Password endpoint allows users to change their password without submitting SSO requests (see the `/ext/pwdchange/Identify` section in */ext/pwdchange/Identify* on page 760). **Note:** The LDAP Username Password Credential Validator (PCV) and the PingOne® Directory 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 data store connection using LDAPS; AD requires this level of security to allow password changes. When connecting to PingDirectory or Oracle Directory Server, configure proxied authorization for the service account on the directory server (see *Configuring proxied authorization* on page 175). This check box 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 check box 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** check box is not selected. This check box is not selected by default.
Enable 'This is My Device'	Allows users to indicate whether their device is shared or private. In this mode, PingFederate authentication sessions (if enabled) are not stored unless the user indicates the device is private. For more information about PingFederate authentication session, see *Sessions* on page 372. This check box 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 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** check box 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 check box is not selected by default.
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. **Note:** This option requires the selection of the **Allow Password Changes** check box. (Both check boxes are not selected by default.) In addition, the LDAP Username PCV is currently the only PCV bundled with PingFederate that supports the password expiring warning feature. This check box is not selected by default.

Field	Description
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 SSPR 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:** It is recommended that the method used is not already part of a multi-factor authentication 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 SSPR 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 SSPR. **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.

Field	Description
Password Reset Policy Contract	If you have selected to use an authentication policy to handle SSPR requests, you must select a policy contract here. Because the sole purpose of this policy contract is for PingFederate to find the applicable authentication policy for password reset, this policy contract does not require any extended attributes. ⓘ **Important:** You must use a policy contract dedicated for password reset. In other words, this policy contract must not be used for SSO anywhere else. If you have not yet defined a policy contract solely for password reset, click **Manage Policy Contracts** to do so.
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 may also provide your users the Account Recovery endpoint shown on the **Summary** screen. The Account Recovery endpoint allows users to unlock their account without submitting SSO requests (see the `/ext/pwdreset/Identify` section in *IdP endpoints*). ⓘ **Note:** You must also select a **Password Reset Type** value other than **None** (and therefore the **Allow Password Changes** check box as well) because the initiating user must prove ownership of the account through the password reset flow. Unlike 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 Active Directory. If the underlying data store is connected to an 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 check box is not selected by default.
Local Identity Profile	Select a local identity profile to offer users the options to authenticate via 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 may also provide your users the Account Recovery endpoint shown on the **Summary** screen. The Account Recovery endpoint allows users to recover their username without submitting SSO requests (see the `/ext/pwdreset/Identify` section in *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 check box is not selected by default.

e. Optional: Click **Show Advanced Fields** to review or modify default values.
f. If you have chosen **Text Message** as the password reset type, click **Manage SMS Provider Settings** to configure the SMS provider through which PingFederate can send text message notifications to the users.

5. On the **Extended Contract** screen, 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 follows:

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 via 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 (see *Enable third-party identity providers* on page 570). Furthermore, regardless of whether 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 (see *Create advanced registration mapping* on page 578).
No selection is made.	The policy.action attribute is not fulfilled.

6. On the **Adapter Attributes** screen, configure the pseudonym and masking options.

   ⓘ **Note:** The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

   a. 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 SP partners use pseudonyms for account linking.

      ⓘ **Note:** A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.
   b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
   c. 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** screen, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for data store queries.
   - Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

HTML Form Adapter advanced fields

Property	Description
Login Template (Required)	The HTML 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.)

Property	**Description**
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 SaaS providers who do not support SAML SLO but want the users' IdP 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 log 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 SP after successful IdP logout (applicable only when the **Logout Path** field is configured). This field has no default value.
Logout Template	The HTML template to be 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 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`.
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 to notify the users that they are being redirected to a password management system to change their password. The default template file is also `html.form.message.template.html`.
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 from the **Notification Publisher** list.
Expiring Password Warning Template	The HTML template to warn the users about approaching the password expiry day. The default template file is `html.form.password.expiring.notification.template.html`.

Property	Description
Threshold for Expiring Password Warning	The threshold (in days) to start warning the user about approaching the password expiry 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).
Login Challenge Template	The HTML 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 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** check box 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** check box is not selected. You may 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** check box on the login form is retained. The lifetime is reset upon each successful login in which the **This is my device** check box on the login form is selected. You may 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 check box 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 check box is not selected by default.

Property	Description
Track Authentication Time	When selected, the time of authentication for each user is tracked and could be utilized by applicable use cases. For example, if an OAuth client sends an authorization request with a max_age parameter, such request will prompt 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 check box 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 should wait prior to the reauthentication attempt. The default value is `0`, which is the minimum value. The maximum value is `60000` (one minute).

Advanced fields for self-service password reset and account unlock

Property	Description
Password Reset Username Template	The HTML template to prompt the user to enter a username for password reset. This template is applicable for all password reset types (other than **None**). The default template file is `forgot-password.html`.
Password Reset Code Template	The HTML template to prompt the user to enter the one-time password (OTP) or the code for password reset. This template is applicable 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 is applicable 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 is applicable for all password reset types (other than **None**). The default template file is `forgot-password-error.html`.
Password Reset Success Template	The HTML template to notify the user that the password reset attempt has succeeded. This template is applicable 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`.
OTP Length	The number of characters in the one-time password for password reset. The default value is `8`.

Property	**Description**
Password Reset Token Validity Time	The validity (in minutes) for the one-time password or the one-time link. The default value is `10` (minutes).
PingID Properties	For self-service password reset using PingID, follow these steps to upload the PingID settings file to the HTML Form Adapter instance: **1.** Sign on to the PingOne admin portal. **2.** Go to the **Setup**# **PingID**# **Client Integration** screen. **3.** Download the settings file (`pingid.properties`). **4.** Close the PingOne admin portal. **5.** Come back to the PingFederate administrative console and upload the `pingid.properties` file to the **PingID Properties** advanced field on the **IdP Adapter** screen.

Advanced fields for self-service username recovery

Property	**Description**
Require Verified Email	When selected, PingFederate only generates notification messages for self-service password reset, account unlock, or username recovery for users who have proven ownership of their email addresses. **Important:** This selection requires that the status of email ownership verification being stored as part of the user record in the directory server and the name of such attribute being the value of the **Mail Verified Attribute** field in the selected LDAP Username PCV instance. The check box is not selected by default.
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 is applicable 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 is applicable 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 containing 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 from the **Notification Publisher** list.

CAPTCHA options

Property	Description
CAPTCHA for Authentication	Enable CAPTCHA to protect the authentication process from automated attacks.
CAPTCHA for Password Change	Enable CAPTCHA to protect the password change process from automated attacks.
CAPTCHA for Password Reset	Enable CAPTCHA to protect the account recovery process (for password reset and account unlock) from automated attacks.
CAPTCHA for Username Recovery	Enable CAPTCHA to protect the username recovery process from automated attacks.

CAPTCHA check boxes are not selected by default.

Kerberos Adapter

The integrated Kerberos Adapter provides a seamless SSO experience for Windows clients by authenticating SSO requests using the Kerberos v5 protocol against Active Directory (AD) domains.

When the PingFederate IdP server receives an authentication request for SP-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 (SIDs) from the ticket, generates a SAML assertion with the username (and optionally the associated domain, SIDs, or both), and passes it to the SP.

Note: The Kerberos Adapter supports authentications by Kerberos only. If your environment requires NTLM support, you must deploy the IWA Integration Kit. You can safely deploy the IWA Adapter and create one or more instances of it alongside with the Kerberos Adapter.

Authentication mechanism assurance

For the purpose of protecting resources based on login 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 logs on using a certificate-based login method, such as a smart-card login. For example, you can restrict access to sensitive resources to users whom log 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 SIDs attribute 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 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 SIDs attribute 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 (technet.microsoft.com/en-us/library/dd378897%28v=ws.10%29.aspx).

Configure a Kerberos Adapter instance

About this task

Steps

1. Click **Identity Provider# Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.
4. On the **IdP Adapter** screen, configure your Kerberos Adapter instance.

   Refer to the on-screen 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 (see *Configuring Active Directory domains or Kerberos realms* on page 209).
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 screen 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 should authentication fail.

Field	Description
Authentication Context Value	This may 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 (see the OASIS documents *oasis-sstc-saml-core-1.1.pdf* 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 As needed, the authentication context can be overridden by 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.)

5. On the **Extended Contract** screen, configure additional attributes for this adapter instance as needed.

   The Kerberos Adapter contract includes three core attributes: Domain/Realm Name, SIDs, and Username.
6. On the **Adapter Attributes** screen, configure the pseudonym and masking options.

   Note: The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

   a. 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 SP partners use pseudonyms for account linking.

      Note: A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.
   b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
   c. 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** screen, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for data store queries.
   - Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

Configure end-user browsers

About this task

You must also configure browsers at your site in order to use the Kerberos Adapter to authenticate users.

> ⓘ **Note:** The client-side configuration requires the base URL or an applicable virtual host name of your PingFederate environment. Base URL is defined on the **System# Protocol Settings# Federation Info** screen. Virtual host names, if configured, are defined on the **System# Virtual Host Names** screen.

> ⓘ **Important:** If the browsers are not properly configured, users may be prompted to authenticate manually using their network credentials or fail to SSO to the service providers.

Configure Microsoft Internet Explorer

About this task

To configure Internet Explorer for Kerberos authentication, review the following settings in Internet Options.

Steps

1. Add the base URL to **Local intranet**.

   > ⓘ **Note:** This step may be skipped if the base URL (<*pf-idp.domain.name*>) is internal and not fully qualified. For example, if it is `pingfederate`, you can skip this step. However, if <*pf-idp.domain.name*> is `www.example.com`, then you must add the base URL to the **Sites** list, as described in the following sub steps.

   a. Close all Internet Explorer tabs and windows.
   b. Open **Control Panel# Internet Options**.
   c. Click the **Security** tab.
   d. Select **Local intranet** and click **Sites**.
   e. Click **Advanced**.
   f. Enter the base URL (for example, `www.example.com`), and then click **Add**.
   g. Click **Close**, and then click **OK** to return to the Security tab.
2. Verify **Automatic logon only in the Intranet zone** is selected.
   a. Under the Security tab, select **Local intranet** and click **Custom level**.
   b. Verify **Automatic logon only in the Intranet zone** is selected in the **Settings** pane.
   c. Click **OK** to return to the Security tab.
3. Verify proxy settings.

   > ⓘ **Note:** Skip the following sub steps if a proxy is not used.

   a. Click the **Connections** tab.
   b. Click **LAN settings**.
   c. Verify the **Use a proxy server for your LAN ...** check box is selected, and then click **Advanced**.
   d. Enter the base URL in the **Exceptions** field, and then click **OK**.
   e. Click **OK** to return to the Connections tab.
4. Verify **Enable Integrated Windows Authentication** is selected.
   a. Click the **Advanced** tab.
   b. Verify **Enable Integrated Windows Authentication** is selected in the **Settings** pane.
5. Click **OK** to close Internet Options.

Configure Mozilla Firefox

About this task

To configure Firefox for Kerberos authentication, configure Firefox as follows:

Steps

1. Start Firefox.
2. Open a new tab, and then enter `about:config` in the address bar.
3. Search for the `network.negotiate-auth.trusted-uris` preference name.
4. Double-click to modify its value to include the base URL of your PingFederate environment (for example, `www.example.com`).
5. Click **OK** and close the `about:config` tab.
6. Optional: Exit Firefox.

OpenToken Adapter

In order to transfer identity and other user information between the PingFederate server and an end application, the product 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 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 has the option to provide an authentication context to the SP by including the authnContext attribute with the desired value in the secure token. Standard URIs are defined in the SAML specifications (see the OASIS documents *oasis-sstc-saml-core-1.1.pdf* and *saml-authn-context-2.0-os.pdf*).

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

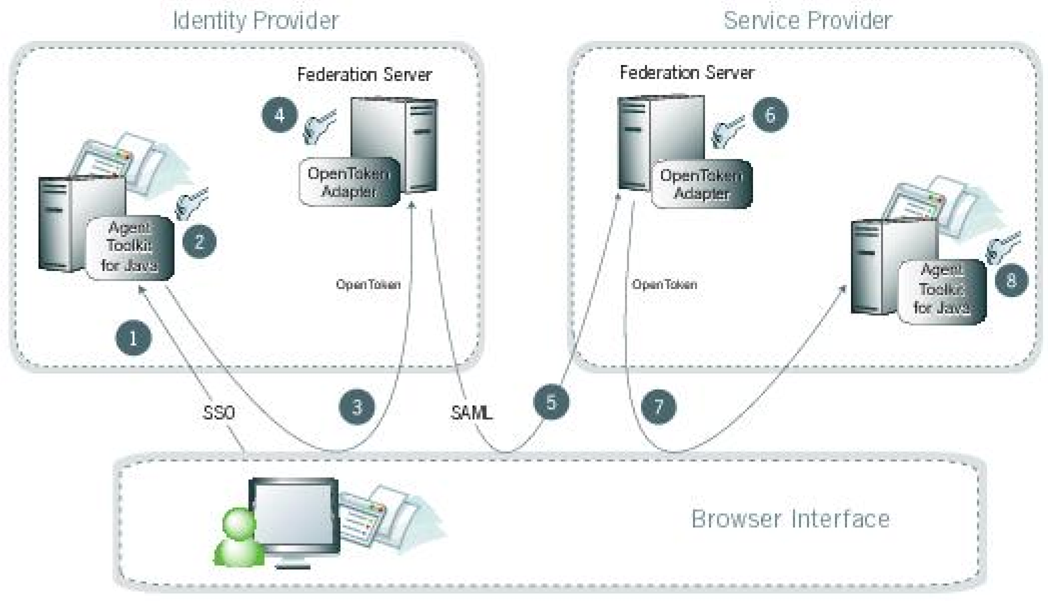
As needed, the authentication context can be overridden by either an instance of the Requested AuthN Context Authentication Selector or the SAML_AUTHN_CTX attribute in the SAML attribute contract. (The latter takes precedence.)

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 this topic to set up the OpenToken Adapter to use with your applications.

The following figure shows a basic IdP-initiated SSO scenario using PingFederate with the Java Integration Kit on both sides of an identity federation.



MY TITLE 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 Security Assertion Markup Language (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.

Configure an OpenToken IdP Adapter instance

About this task

Steps

1. Click **Identity Provider# Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.

4. On the **IdP Adapter** screen, configure your OpenToken IdP Adapter instance.

> **Note:** These values are dependent on your developer's implementation.

Refer to the on-screen field descriptions and the following table for more information.

Field	Description
Password Confirm Password (Required)	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 an SSO event. This URL is part of an external application, which performs user authentication.
Click **Show Advanced Fields** to review the following settings. Modify as needed.	
Transport Mode	How the token is transported to and from the application, either via 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 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 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.

Field	Description
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. The check box is not selected by default. **Note:** Selecting this option may 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. We recommend to update your applications with the latest version of the agent. We recommend not to change the value of this flag.

5. On the **Actions** screen, 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 IdP application. Refer to the documentation of your respective integration kit for more information.
6. On the **Extended Contract** screen, configure additional attributes for this adapter instance as needed.

   The OpenToken IdP Adapter contract includes one core attribute: subject.

   Note that 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** screen, configure the pseudonym and masking options.

   Note: The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

   a. 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 SP partners use pseudonyms for account linking.

      Note: A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.
   b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
   c. 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

8. Optional: On the **Adapter Contract Mapping** screen, configure the adapter contract for this instance with the following optional workflows:
   - Configure one or more data sources for data store queries.
   - Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
10. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

    If you want to exit without saving the configuration, click **Cancel**.

Configure an OpenToken SP Adapter instance

About this task

Steps

1. Click **Service Provider**# **Adapters** to open the **Manage SP Adapter Instances** screen.
2. On the **Manage SP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.
4. On the **Instance Configuration** screen, configure your OpenToken SP Adapter instance.

   Note: These values are dependent on your developer's implementation.

   Refer to the on-screen field descriptions and the following table for more information.

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** to review the following settings. Modify as needed.	
Transport Mode	How the token is transported to and from the application, either via 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**.

Field	Description
Authentication Service	The URL to which the user is redirected for an 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.
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 checked (the default), it prevents insecure content from affecting the security of your application and the agent. We recommend to update your applications with the latest version of the agent. We recommend not to change the value of this flag.

Field	Description
URL Encode Cookie Values	If checked, the extended attribute cookie value will be URL encoded.

5. On the **Actions** screen, 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 SP application. Refer to the documentation of your respective integration kit for more information.
6. Optional: On the **Extended Contract** screen, configure additional attributes for this adapter instance.
7. On the **Summary** screen, review your configuration, modify as needed, and click **Done**.
8. On the **Manage SP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

Composite Adapter

For an IdP, PingFederate includes a Composite Adapter, which 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 may be used to choose an adapter instance based on the method by which a user authenticated, or to integrate an organization's multifactor authentication requirement.

Tip: For complex authentication requirements, consider implementing one or more authentication policies on the **Policies** screen from the **Identity Provider** or **Service Provider** menu.

Configure a Composite Adapter instance

About this task

Steps

1. Click **Identity Provider#** **Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.

4. On the **IdP Adapter** screen, configure your Composite Adapter instance as follows:

a. Click **Add a new row to 'Adapters'**.

b. Select an IdP adapter instance from the **Adapter Instance** list, configure the selected adapter instance and then click **Update**.

Refer to the on-screen field descriptions and the following table for more information.

Column	Description
Policy (Required)	**Required** (the default) indicates authentication via this adapter instance is needed to continue SSO processing and to invoke any remaining instances in the chain. If you are integrating multifactor 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 via 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 may be returned from the adapter instance. The value in this field may 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.

c. 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: Several types of adapters—for example, the IWA IdP Adapter—may be configured to direct end users to an error page if authentication fails for any reason, which will halt further progress through a composite-adapter chain. Ensure that for such adapter instances the **Error URL** option is *not* used in the instance configuration, if continuation through an adapter-chaining sequence is required.

d. Optional: Use the workflow under **Action** to manage the selected adapter instances.

e. 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 screen. 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. Select the HTML Form Adapter instance from the **Target Adapter** list.
3. Select a source attribute from the **User ID Selection** list.
4. Click **Update**.

Note: For OAuth use cases, entries in the **Input User ID Mapping** section could 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.

f. 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'** and 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.

g. Defines 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 (**Add to Back**) the 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 may 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** screen, **Add** attributes to be returned from each adapter instance configured on the previous screen.

Note: Attributes must correspond exactly to any or all of the attribute names listed on the Adapter Attribute screens for each configured adapter instance.

6. On the **Adapter Attributes** screen, configure the pseudonym and masking options.

   > **Note:** The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

   a. 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 SP partners use pseudonyms for account linking.

      > **Note:** A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.

   b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
   c. 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** screen, configure the adapter contract for this instance with the following optional workflows:
   - Configure one or more data sources for data store queries.
   - Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

HTTP Basic Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or an IdM 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* (PCV); for example, an LDAP Username PCV. This authentication mechanism validates credentials against a user repository via an instance of a PCV. Multiple PCV instances may be added 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*. If the HTTP Basic Adapter does not finds 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

As needed, the authentication context can be overridden by 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.)

Configure an HTTP Basic Adapter instance

About this task

Steps

1. Click **Identity Provider# Adapters** to open the **Manage IdP Adapter Instances** screen.
2. On the **Manage IdP Adapter Instances** screen, click **Create New Instance** to start the **Create Adapter Instance** configuration wizard.
3. On the **Type** screen, configure the basics of this adapter instance.
   a. Enter the required information and select the adapter type from the list.
   b. Optional: 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 screens.
4. On the **IdP Adapter** screen, configure your HTTP Basic Adapter instance as follows:
   a. If you have not yet defined the desired Password Credential Validator instance, click **Manage Password Credential Validators** to do so.
   b. Click **Add a new row to 'Credential Validators'** to select a credential-authentication mechanism instance for this adapter instance.
   c. Select a Password Credential Validator instance from the list and 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 is able to 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 lockout those accounts in their source locations.
   d. Enter values for the adapter configuration.

      Refer to the on-screen field descriptions and the following table for more information.

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:** Once 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`.

5. On the **Extended Contract** screen, configure additional attributes for this adapter instance as needed.

   The HTTP Basic Adapter contract includes one core attribute: username.

6. On the **Adapter Attributes** screen, configure the pseudonym and masking options.

   > **Note:** The **Override Attributes** check box in this screen reflects the status of the override option in the **Extended Contract** screen.

   a. 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 SP partners use pseudonyms for account linking.

      > **Note:** A selection is required regardless of whether 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 (for example, email) to prevent the same pseudonym from being assigned to multiple users.

   b. Select the check box under **Mask Log Values** for any attributes that you want PingFederate to mask their values in its logs at runtime.
   c. 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** screen, configure the adapter contract for this instance with the following optional workflows:

   - Configure one or more data sources for data store queries.
   - Fulfill adapter contract with values from the adapter (the default), data store 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** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

Self-service user account management

PingFederate provides various self-service applications for end users to manage their accounts. These optional capabilities lower the costs of identity management, 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 screens, the user-facing templates can be customized and localized to provide the desired user experience.

Refer to subsequent topics for configuration steps.

> **Tip:** Similarly, when configuring customer identity and access management use cases, administrators can enable the end users to manage their local account, connect or disconnect one or more social connections, and change or set the password for their local account. For more information, see *Customer IAM configuration* on page 558 and *Enable profile management* on page 576.

Configure self-service password management

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 LDAP Username Password Credential Validator (PCV). You may configure PingFederate to generate notification messages when users have successfully changed 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® Directory PCV, you can also enable the change password capability. Note that notifications for change password and password expiry are not supported at this point.

Steps

1. On the **System**# **Data Stores** screen, create a new LDAP data store.

   You can also reuse an existing LDAP data store connection.

   Important: When connecting to an Active Directory (AD) LDAP server, you *must* secure the data store connection using LDAPS; AD requires this level of security to allow password changes.

   When connecting to PingDirectory or Oracle Directory Server, configure proxied authorization for the service account on the directory server (see *Configuring proxied authorization* on page 175).

   This step does not apply if you are validating credentials through the PingOne Directory PCV.
2. On the **System**# **Password Credential Validators** screen, create a new instance of the LDAP Username PCV or the PingOne Directory PCV.

   You can also reuse an existing LDAP Username PCV or PingOne Directoryinstance.

   If you are validating credentials through the LDAP Username PCV and if you want to enable notifications, skip to *step 3b* to configure the related advanced fields.

   a. Select a data store, enter a search base, define a search filter, select the scope of search, and enable or disable case-sensitive matching.
   b. Click **Show Advanced Fields** to update fields related to notifications.

      Configuration items vary depending on your requirements and the directory setup. Refer to the following table for more information.

Field	Description
Display Name Attribute	The LDAP attribute that PingFederate uses to personalize the notification message. The default value is `displayName`.
Mail Attribute	The LDAP attribute containing the email address that PingFederate uses as the destination of the notification message. The default value is `mail`.

3. On the **Identity Provider# Adapters** screen, create a new HTML Form Adapter instance.

   You can also reuse an existing HTML Form Adapter instance. If so, skip to *step 4c* to configure your adapter instance to enable self-service password management.

   a. Select the PCV instance defined in the previous step as the credential validator.
   b. Optional: Update any default values or options.
   c. Select the **Allow Password Changes** check box.
   d. Select the **Change Password Notification** check box if you want PingFederate to generate a notification message 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.
   e. Select the **Show Password Expiring Warning** check box if you want the **Sign On** screen to warn the user about an approaching password expiration.
   f. Select a notification publisher instance from the list if you have selected the **Change Password Notification** check box.

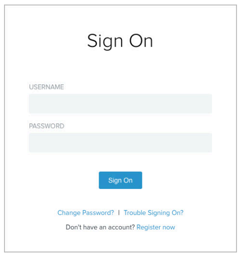
      If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.
   g. Click **Show Advanced Fields** to review or modify default values related to the change password capability.

      For example, update the **Change Password Template** field if you want to use a custom template to render the **Change Password** screen.
4. Optional: Customize and localize the on-screen messages and notification messages.

Results

You have now successfully 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, as illustrated in this screen capture.


Sign On
USERNAME
PASSWORD
Sign On
Change Password? | Trouble Signing On?
Don't have an account? Register now

Additionally, 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 SSO requests.

Configure self-service account recovery

About this task

PingFederate offers self-service password reset (SSPR) for users to recover their account in the event of forgotten password. Integrated into the HTML Form Adapter and Password Credential Validator (PCV) framework, users can now reset their password via one of the following mechanisms:

- Authentication policy
- One-time link via email

- One-time password via email
- One-time password via text message
- PingID®

The SSPR capability relies on the HTML Form Adapter and the LDAP Username PCV to query the required attributes for the chosen reset mechanism. PingFederate supports PingDirectory™ , Microsoft Active Directory, and Oracle Directory Server out-of-the-box. Custom PCV implementations may also be developed to offer the SSPR features for users stored in non-LDAP data sources. For more information, refer to the `ResettablePasswordCredential` interface in Javadoc.

> **Tip:** The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.

PingFederate also provides the capability for users to unlock their account 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** screen 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.

Unlike password reset, 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 Active Directory. If the underlying data store is connected to an Oracle Directory Server, users can only unlock their account by changing their current password through the password reset flow.

Steps

1. On the **System**# **Data Stores** screen, create a new LDAP data store.

   You can also reuse an existing LDAP data store connection.

   > **Important:** When connecting to an Active Directory (AD) LDAP server, you *must* secure the data store connection using LDAPS; AD requires this level of security to allow password changes.
   >
   > When connecting to PingDirectory or Oracle Directory Server, configure proxied authorization for the service account on the directory server (see *Configuring proxied authorization* on page 175).
   >
   > When connecting to PingDirectory, configure the account usability control ACI for the service account on the directory server if you intend to enable self-service account unlock (see *Configuring the account usability control ACI* on page 175).

2. On the **System**# **Password Credential Validators** screen, create a new LDAP Username PCV instance.

   You can also reuse an existing LDAP Username PCV instance. If so, skip to *step 3b* to configure the related advanced fields.

   a. Select a data store, enter a search base, define a search filter, select the scope of search, and enable or disable case-sensitive matching.
   b. Click **Show Advanced Fields** to update fields related to SSPR.

      Configuration items vary depending on the desired password reset type and the directory setup. Refer to the following table for more information.

Field	Description
Display Name Attribute	The LDAP attribute that PingFederate uses to personalize the notification message. The default value is `displayName`.

Field	Description
Mail Attribute	The LDAP attribute containing the email address that PingFederate uses as the destination of the notification message. This field is required when the password reset type is **Email One-Time Link** or **Email One-Time Password** in any invoking HTML Form Adapter instances. The default value is `mail`.
SMS Attribute	The LDAP attribute containing the phone number, to which PingFederate sends text message notifications to the requesting users. This field is required when the password reset type is **Text Message** in any invoking HTML Form Adapter instances. There is no default value.
PingID Username Attribute	The LDAP attribute containing the username to use for PingID based password reset. This field is required when the password reset type is **PingID** in any invoking HTML Form Adapter instances (see *step 4e*). There is no default value.

3. On the **Identity Provider# Adapters** screen, create a new HTML Form Adapter instance.

   You can also reuse an existing HTML Form Adapter instance. If so, skip to *step 4c* to configure your adapter instance to enable the SSPR and account unlock capabilities.

   a. Select the LDAP Username PCV instance defined in the previous step as the credential validator.
   b. Optional: Update any default values or options.
   c. Select the **Allow Password Changes** check box.
   d. Select the **Change Password Notification** check box if you want PingFederate to generate a notification message 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.
   e. Select the desired password reset type.

Field	Description
Password Reset Type	Select one of the following methods for SSPR. **Authentication Policy** Based on the policy contract selected from the **Password Reset Policy Contract** list, PingFederate finds the applicable authentication policy to handle SSPR 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:** It is recommended that the method used is not already part of a multi-factor authentication 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 SSPR 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 SSPR. **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**.

Field	**Description**
Password Reset Policy Contract	If you have selected to use an authentication policy to handle SSPR requests, you must select a policy contract here. Because the sole purpose of this policy contract is for PingFederate to find the applicable authentication policy for password reset, this policy contract does not require any extended attributes. **Important:** You must use a policy contract dedicated for password reset. In other words, this policy contract must not be used for SSO anywhere else. If you have not yet defined a policy contract solely for password reset, click **Manage Policy Contracts** to do so.

f. Select the **Account Unlock** check box if you want to enable self-service account unlock as well.
g. Select a notification publisher instance from the list.

   If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.
h. Click **Show Advanced Fields** to review or modify the rest of the default values related to SSPR.

   For information regarding the **PingID Properties** field, refer to the following table.

Field	**Description**
PingID Properties	For self-service password reset using PingID, follow these steps to upload the PingID settings file to the HTML Form Adapter instance: **1.** Sign on to the PingOne admin portal. **2.** Go to the **Setup**# **PingID**# **Client Integration** screen. **3.** Download the settings file (`pingid.properties`). **4.** Close the PingOne admin portal. **5.** Come back to the PingFederate administrative console and upload the `pingid.properties` file to the **PingID Properties** advanced field on the **IdP Adapter** screen.

4. If you have selected **Authentication Policy** as the password reset type, create a new authentication policy to handle SSPR requests.

Generally speaking, a password reset policy must authenticate users through means other than prompting for the forgotten passwords. It should also enforce multi-factor authentication for added security. For illustration, 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 multi-factor authentication. If users satisfy both authentication requirements, the policy uses a policy contract to relay user attributes to partners. (To learn more about this policy configuration, see *Defining authentication policies based on group membership information* on page 347.)

Like SSO, you also want to protect SSPR with multi-factor authentication.

Knowing your company actively manages client certificates on company devices, you have decided to use an instance of the X.509 IdP Adapter (named X.509) as the first-factor authentication source in your password reset policy. You have 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 4e*, 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.

a. On the **Identity Provider# Policies** screen, click **Add Policy**.
b. On the **Policy** screen, enter a name (and optionally a description) for the policy.
c. Select the X.509 IdP Adapter instance.
d. Configure each policy path out of the X.509 Adapter instance.

   Fail

   Select **Done**, which terminates the SSPR request.

   For instance, if a user submits an SSPR 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.

e. Configure 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.

   For more information, see *Specifying an incoming user ID* on page 344.
f. Configure each policy path out of the PingID Adapter instance.

   Fail

   Select **Done**, which terminates the SSPR request.

   Success

   Select **SSPR APC**, which is the policy contract created solely for password reset per *step 4e*.

   Important: You must not reuse this policy contract for SSO elsewhere.

g. Configure 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

h. Click **Done** and then **Save**.

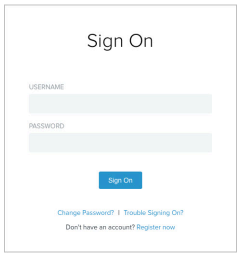
This sample use case demonstrates the capability and flexibility that a password reset policy offers. Depending on actual use cases, you may use a different series of authentication sources to authenticate users in a secure manner. For instance, if your organization manages devices using AirWatch, you may 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.

Results

You have now 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, as illustrated in this screen capture.


Sign On
USERNAME
PASSWORD
Sign On
Change Password? | Trouble Signing On?
Don't have an account? Register now

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.

Configure self-service username recovery

About this task

PingFederate offers self-service username recovery for users to recover their account in the event of forgotten username via email.

When enabled, a user who forgot the username can recover it by providing an email address. If PingFederate can locate the user record using such email address, PingFederate sends to the user at the provided address an email message containing the recovered username. If the email ownership verification status is stored as part of the user record in the directory server, it is also possible to restrict the delivery of username recovery email messages to users who have proven ownership of their email addresses.

This optional capability is integrated into the HTML Form Adapter and the LDAP Username Password Credential Validator (PCV). PingFederate supports PingDirectory™ , Microsoft Active Directory, and Oracle Directory Server out-of-the-box. Custom PCV implementations may also be developed to offer the same capability for users stored in non-LDAP data sources. For more information, refer to the `RecoverableUsername` interface in Javadoc.

Tip: The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.

Steps

1. On the **System**# **Data Stores** screen, create a new LDAP data store.

   You can also reuse an existing LDAP data store connection.

2. On the **System#** **Password Credential Validators** screen, create a new LDAP Username PCV instance.

You can also reuse an existing LDAP Username PCV instance. If so, skip to *step 3b* to configure the related advanced fields.

a. Select a data store, enter a search base, define a search filter, select the scope of search, and enable or disable case-sensitive matching.
b. Click **Show Advanced Fields** to update fields related to self-service username recovery.

Configuration items vary depending on your requirements and the directory setup. Refer to the following table for more information.

Field	Description
Display Name Attribute	The LDAP attribute that PingFederate uses to personalize the notification message. The default value is `displayName`.
Mail Search Filter (for username recovery)	The LDAP query to locate a user record using an email address; for example: `mail=${mail}` ⓘ **Note:** When configuring in conjunction with password reset, the attribute specified in the left side of this search filter should correspond to the attribute specified in the **Mail Attribute** field. There is no default value.
Username Attribute (for username recovery)	The LDAP attribute containing the user identifier of the users. ⓘ **Note:** This attribute should correspond to the attribute specified in the left side of the **Search Filter** field. There is no default value.
Mail Verified Attribute (for username recovery)	The LDAP attribute indicating whether the user's email address has been verified. The expected value of this user attribute must either be `true` or `false` (case insensitive). This field is required if the HTML Form Adapter instance is configured to only generate username recovery notification messages to users who have proven ownership of their email addresses (see this *sub step*). There is no default value.

3. On the **Identity Provider# Adapters** screen, create a new HTML Form Adapter instance.

   You can also reuse an existing HTML Form Adapter instance. If so, skip to *step 4c* to configure your adapter instance to enable the self-service username recovery capability.

   a. Select the LDAP Username PCV instance defined in the previous step as the credential validator.
   b. Optional: Update any default values or options.
   c. Select the **Enable Username Recovery** check box.
   d. Select a notification publisher instance from the list.

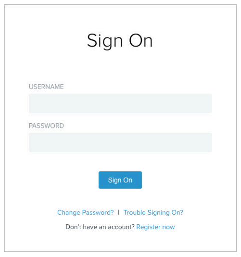
      If you have not yet configured the desired notification publisher instance, click **Manage Notification Publishers**.
   e. Click **Show Advanced Fields** to review or modify default values related to self-service username recovery.

      For example, select the **Require Verified Email** check box if you want PingFederate to only send username recovery email messages to users who have proven ownership of their email addresses.
4. Optional: Customize and localize the on-screen messages and notification messages.

Results

You have now successfully created a new instance or modified an existing instance of the HTML Form Adapter with the self-service username recovery capability.

When a user signs on through this adapter instance, the user has the option to recover the username using the **Trouble Signing On** link, as illustrated in this screen capture.


Sign On
USERNAME
PASSWORD
Sign On
Change Password? | Trouble Signing On?
Don't have an account? Register now

Additionally, you can also provide your users the per-adapter Account Recovery endpoint (`/ext/pwdreset/Identify`), which allows them to recover their username through this HTML Form Adapter instance without submitting SSO requests.

Application endpoints

These endpoints provide a means, via standard HTTP, by which external applications can communicate with the PingFederate server.

The SSO and SLO endpoints for an IdP and an 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 (see *Customizable user-facing screens* on page 257).

SP endpoints also include those available for SCIM inbound provisioning (see *Provisioning for SPs* on page 126).

For either SP or IdP servers, a maintenance endpoint is also provided for administrators to verify that the server is running. Endpoints applicable to both server roles also include those needed for adapter-to-adapter mapping (see *Adapter-to-adapter mappings* on page 701) and retrieval of WS-Trust metadata (see *WSC and WSP support* on page 101).

PingFederate provides a favorite icon for all Application Endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 291.

IdP endpoints

The following sections describe PingFederate 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 the parameter TargetResource (or TARGET) is used and includes its own query parameters, the parameter value must be URL-encoded.

Any other parameters that contain restricted characters (many SAML URNs, for example) also must be URL-encoded.

For information about URL encoding, please refer to third party resources, such as HTML URL-encoding Reference (www.w3schools.com/tags/ref_urlencode.asp).

User-facing templates can be customized and localized.

/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 an 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* on page 817.

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. Note that 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 (see *Set Assertion Consumer Service URLs (SAML)* on page 523). 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 may be overridden by policy based on authentication policies. For example, an CIDR Authentication Selector instance could enforce the use of a given adapter instance based on whether a user is on or off the network (see *Authentication policies* on page 324).
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:** In order to use this parameter, the **Allow Password Changes** check box must be selected in the adapter configuration of the invoked HTML Form Adapter instance (see).
RequestedFormat (optional - SAML 2.0)	Allows control over the NameId format.
vsid (optional)	Specify the virtual server ID. When absent, PingFederate uses the default virtual server ID (if specified) for the connection (see *Identify the SP* on page 504) or the SAML federation ID defined in Server Settings (see *Specifying federation information* on page 137).

/idp/startSLO.ping

This is the path used to initiate an IdP-initiated SLO (under SAML 2.0) or an OpenID Connect logout (see *Asynchronous Front-Channel Logout* on page 483). 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.

The following table shows 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 screen. Note that 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 be used; 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 (see *Specify SLO service URLs (SAML 2.0)* on page 526).

/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. Note that 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 screen.

/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 screen.

/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**# **Redirect Validation (Local Redirect Validation)** screen, PingFederate honors the URL only if the parameter value satisfies the configured requirement on the **Redirect Validation (Local Redirect Validation)** screen. If the validation fails, PingFederate displays the default success or error message. (For more information, see *Configure redirect validation* on page 316.)

/ext/pwdreset/Identify

The Account Recovery endpoint allows users to reset their password, unlock their account, or 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 are 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, unlock their account, or recover their username. ⓘ **Note:** When target resource validation is enabled for this endpoint, as indicated by the **SLO and Other** check box on the **Security# Redirect Validation (Local Redirect Validation)** screen, PingFederate honors the URL only if the parameter value satisfies the configured requirement on the **Redirect Validation (Local Redirect Validation)** screen. If the validation fails, PingFederate displays the default success or error message. (For more information, see *Configure redirect validation* on page 316.)

SP endpoints

The following sections describe the PingFederate SP endpoints for SP services and SCIM inbound provisioning.

SP services

The following sections describe PingFederate 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 the parameter TargetResource (or TARGET) is used and includes its own query parameters, the parameter value must be URL-encoded.

Any other parameters that contain restricted characters (many SAML URNs, for example) also must be URL-encoded.

For information about URL encoding, please refer to third party resources, such as HTML URL-encoding Reference (www.w3schools.com/tags/ref_urlencode.asp).

Parameters are case-sensitive.

/sp/startSSO.ping

This is the path used to initiate SP-initiated SSO. In this scenario, the SP issues an SSO request to the IdP asking for an SSO authentication response. Typically, a systems integrator or developer creates one or more links to this endpoint in SP applications to allow users to access various protected resources via 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* on page 817.

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.
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 is able to 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 where the end-user is redirected after a successful SSO (the target applications). Note that the parameter value must be URL-encoded. When this parameter is not provided in the URL, a default target resource may be specified in the administrative console, either for all IdP connections (see *Configure default URLs* on page 589) or for individual connections (see *Configure default target URLs* on page 622), or both.
InErrorResource (optional)	This parameter indicates where the end-user is redirected after an unsuccessful SSO. If this parameter is not included in the request, PingFederate redirects the user to the SLO error landing page hosted within PingFederate (see *Customizable user-facing screens* on page 257).

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`
	`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 (see *Specify SSO service URLs (SAML)* on page 614).

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 (see section 3.4.1.3.1 of the OASIS SAML document *saml-core-2.0-os.pdf*). Multiple values are permitted in order to build a preferred list.

Parameter	Description
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, please refer to the *OpenID Connect specification* (openid.net/specs/openid-connect-core-1_0.html#AuthRequest). The default is `false`.
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 (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. Multiple values are permitted in order 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 (see section 2.7.2.2 of the OASIS SAML document *saml-core-2.0-os.pdf*). Multiple values are permitted in order 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 attribute is not included in the AuthnRequest.
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, see *SP affiliations* on page 556).
vsid (optional)	Specify the virtual server ID. When absent, PingFederate uses the default virtual server ID (if specified) for the connection (see *Identify the partner* on page 599) or the SAML federation ID defined in Server Settings (see *Specifying federation information* on page 137).

If an adapter is specified in SpSessionAuthnAdapterId, 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 screen (see *Configure target URL mapping* on page 586).

Note that adapter selection for SP-initiated SSO is similar to that for IdP-initiated SSO except that, because the adapter ID is dependent 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 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 SP Default URLs screen. Note that the parameter value must be URL-encoded.

Binding
 	
(optional - SAML 2.0)
	Indicates the binding to be used; 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 (see *Specify SLO service URLs (SAML 2.0)* on page 526).

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 (see *Customizable user-facing screens* on page 257).

/sp/defederate.ping

This is the path used to terminate an account link created during SSO. Account linking provides a means for subject identification on the SP side. Links are created and terminated entirely by a user on the SP side. 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 via account linking on the same PingFederate session, each of those links will be terminated by this endpoint. A local logout is also performed for any link that is terminated.

/sp/cdcstartSSO.ping

This endpoint is used for IdP-Discovery implementations (see *Standard IdP Discovery* on page 45). 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 is used to initiate an Attribute Query with a SAML 2.0 IdP (see *Attribute Query and XASP* on page 45).

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 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. If this parameter is not present, 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 (see section 8.3 of the *SAML specifications* (docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf). **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`. Note that the parameter must be URL-encoded.
AppId	The unique identifier of the initiating application.

Parameter	Description
SharedSecret	Used to authenticate the initiating application. Both the AppId and SharedSecret values must match those defined on the **Security# Service Authentication** screen. ⓘ **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
RequestedAttrName (optional)	A name of a user attribute requested from the IdP. For each such desired user attribute, include this parameter. If this parameter is not present, then all allowable user attributes are returned from the IdP. Multiple values are permitted in order 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 (see *Identify the partner* on page 599) or the SAML federation ID defined in Server Settings (see *Specifying federation information* on page 137).

SCIM inbound provisioning endpoints

PingFederate supports SCIM inbound provisioning and provides four endpoints:

- `/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* (www.simplecloud.info/specs/draft-scim-core-schema-01.html)
- *SCIM Specification* (www.simplecloud.info/specs/draft-scim-api-01.html)

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 data store. 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.)

HTTP method	Description

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`*
	▪ Retrieve 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 (www.simplecloud.info/specs/draft-scim-api-01.html#query-resources).
	`/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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.1).
	`/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. Note that, depending on the implementation of the target data store, the target data store may 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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.2).
	`/pf-scim/v1/Users?startIndex=`*`x`*`[&count=`*`y`*`]`
	▪ 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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.3).
	Tip: You may use a combination of the aforementioned parameters in one query to narrow your search results.

HTTP method	**Description**
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 data store. ▪ 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** screen in the applicable IdP connection. ▪ A successful response is indicated by an HTTP 200 status code.

Note: For a list of HTTP error codes that may be returned, see *3.9 HTTP Response Codes* in SCIM Specification (www.simplecloud.info/specs/draft-scim-api-01.html#anchor6).

/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** screen 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 data store. 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.)

HTTP method	Description

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 (www.simplecloud.info/specs/draft-scim-api-01.html#query-resources).
	`/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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.1).
	`/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. Note that, depending on the implementation of the target data store, the target data store may 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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.2).
	`/pf-scim/v1/Groups?startIndex=`*`x`*`[&count=`*`y`*`]`
	▪ 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 (www.simplecloud.info/specs/draft-scim-api-01.html#rfc.section.3.2.2.3).
	Tip: You may use a combination of the aforementioned parameters in one query to narrow your search results.

HTTP method	Description
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 data store. ▪ 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 may be returned, see *3.9 HTTP Response Codes* in SCIM Specification (www.simplecloud.info/specs/draft-scim-api-01.html#anchor6).

/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.

Sample output

```
$ 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 SP configuration endpoint is where developers can retrieve detailed information on the PingFederate SCIM 1.1 implementation. When inbound provisioning is enabled 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.

Sample output

```
$ 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

These endpoints apply to the PingFederate server generally, whether used as an IdP, SP, or both.

Note: 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 (see *Customize the heartbeat message* on page 290).

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 (see *Runtime reporting* on page 159).

/pf/adapter2adapter.ping

This endpoint initiates direct IdP-to-SP adapter mapping, when that feature is configured on the **Adapter-to-Adapter Mappings** screen (see *Adapter-to-adapter mappings* on page 701).

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 *Configure 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 on the **Service Provider**# **Default URLs** screen.
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 (see *Customizable user-facing screens* on page 257).
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 entries defined on the **Service Provider**# **Target URL Mapping** screen (see *Configure target URL mapping* on page 586).

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:** In order to use this parameter, the **Allow Password Changes** check box must be selected in the adapter configuration of the invoked HTML Form Adapter instance (see).

/pf/sts.wst

This endpoint 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 on the **Token Translator Mappings** screen (see *Token translator mappings* on page 705).

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, a secondary PingFederate listening port must be configured and used by partners or STS clients for the relevant endpoints—`*.ssaml*` and `*.wst` (see *Configuring PingFederate properties* on page 221).

/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	Specify the virtual server ID.
(optional)	If absent, PingFederate uses the default virtual server ID (if specified) for the connection or the federation ID defined on the **System# Protocol Settings# Federation Info** screen.

ⓘ **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 .

/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	Specify the virtual server ID.
(optional)	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 in the **System# Protocol Settings# Federation Info** screen.

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 .

Construct an alternative metadata exchange endpoint

About this task

You can embed virtual server ID information into an STS metadata exchange endpoint or a SAML and WS-Federation metadata exchange endpoint. This is useful for scenarios where partners prefer to retrieve metadata by sending one query parameter (PartnerSpId or PartnerIdpId) instead of two query parameters (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>"}`

   For example, if the virtual server ID is `Engineering`, the JSON object is:

   `{"vsid":"Engineering"}`
2. Base64url-encode the JSON object.

   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/sts_mex.ping`)

   Between `/pf` and `/federation_metadata.ping`

STS metadata endpoint (`/pf/sts_mex.ping`)

Between `/pf` and `/sts_mex.ping`

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

OAuth 2.0 endpoints

When developing OAuth-capable applications, developers must follow the OAuth 2.0 Authorization Framework and OpenID Connect specifications (if applicable), which means that the applications must send requests to various OAuth endpoints to obtain authorization grants, access tokens, and (if applicable) refresh tokens and ID tokens. Furthermore, there are additional endpoints for other purposes, such as clients to validate access and refresh tokens, developers to submit client registrations using the OAuth 2.0 Dynamic Client Registration protocol, clients to retrieve OpenID Connect metadata, and more.

Each endpoint extends from the runtime server at the base URL. If virtual host names are configured, the endpoints are also accessible at those locations as well.

For example, if the base URL is https://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

- https://www.example.com:9031/as/authorization.oauth2
- https://www.example.org:9031/as/authorization.oauth2
- https://www.example.info:9031/as/authorization.oauth2

Token endpoint /as/token.oauth2

- https://www.example.com:9031/as/token.oauth2
- https://www.example.org:9031/as/token.oauth2
- https://www.example.info:9031/as/token.oauth2

The subsequent topics describe each endpoint in detail. Unless otherwise indicated, these endpoints and associated parameters are defined in the OAuth and OpenID Connect specifications.

Authorization endpoint

The authorization endpoint is defined in *the OAuth 2.0 Authorization Framework* (tools.ietf.org/html/rfc6749#section-3.1) and is used by the OAuth AS to interact directly with resource owners, authenticate them, and obtain their authorizations. Typically, an OAuth client makes an authorization request by directing a resource owner, via an HTTP user-agent, to the authorization endpoint. After completing 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 may be used as part of 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.

ⓘ **Note:** This endpoint accepts the HTTP GET and POST methods.

Endpoint: /as/authorization.oauth2

The following table describes parameters for this endpoint. The required Content-Type value is `application/x-www-form-urlencoded` when transmitting via the HTTP POST method.

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* (openid.net/specs/oauth-v2-form-post-response-mode-1_0.html).
response_type (Required)	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.
code_challenge	Supply a one-time string value to associate the authorization request with the token request to reduce the risk of authorization code interception attack. For more information, refer to the *Proof Key for Code Exchange (PKCE) by OAuth Public Clients* specification (tools.ietf.org/html/rfc7636). Applicable only when response_type parameter value is `code`. Mandatory if the client is required to so (see the **Require Proof Key for Code Exchange (PKCE)** setting on *Configure an OAuth client* on page 416). ⓘ **Note:** If used, the OAuth client must submit the corresponding code verifier when using the authorization code to obtain an access token (see code_verifier in *OAuth grant type parameters* on page 792).

Parameter	**Description**
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: `code_challenge=Base64Url-encode(SHA256(ASCII(code_verifier)))`, where: ▪ `ASCII(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, refer to *Proof Key for Code Exchange (PKCE) by OAuth Public Clients* (tools.ietf.org/html/rfc7636). The code_challenge_method parameter value is case-sensitive. An error message is returned to the clients for any other values. Note that omitting the code_challenge_method parameter has the same effect as providing the code_challenge_method parameter with a value of `plain`.
redirect_uri	The URI to which PingFederate redirects the resource owner's user-agent after an authorization is obtained. For OpenID Connect 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 all 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. However, for clients that have been configured with only one redirection URI, this requirement is waived to minimize the impact that it may 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 (see *OAuth grant type parameters* on page 792).

Parameter	Description
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 guidelines defined under *IETF BCP 47* (tools.ietf.org/html/bcp47). ⓘ **Tip:** You can map the claims_locales value into the persistent grants (and therefore the access tokens, the ID tokens, or both) from an IdP adapter or an IdP connection by selecting **Context** under **Source** and **Requested Claims Locales** under **Value** in the **Contract Fulfillment** screen 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	The allowable elapsed time (in seconds) since the end users last authenticated. If the elapsed time exceeds the value of max_age, the end users are prompted for reauthentication. ⓘ **Tip:** The HTML Form Adapter supports the max_age parameter by tracking the authentication time for each user.

Parameter	Description
request	A single, self-contained parameter; a signed JWT whose claims represent the request parameters of the authorization request. The OpenID Connect specification calls this JWT a *request object*. 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 algorithm (or algorithms). If the signature does not pass the verification process, the request fails. Optional if a client is not configured to transmit request parameters in signed request objects but it 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 algorithm (or algorithms). If the signature does not pass the verification process, the request fails. Ignored if a client is not configured to transmit request parameters in signed request objects and it is not configured with a JWKs URL or an actual JWKs. When PingFederate receives an authorization request with a signed request object, it processes the authorization request without taking the signed request object into consideration. 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. **Note:** If a client includes in an authorization request a request parameter (other than client_id and response_type) 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 and response_type request parameters, 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. It is also worth noting that if a request parameter is found only outside of the signed request object, such request parameter is dropped and ignored; no error message is returned. **Tip:** Per OAuth and OpenID Connect specifications, a client must always include in an authorization request the client_id, response_type, and scope request parameters outside of the signed request object. For client configuration information, see *Configure an OAuth client* on page 416 (the **Require Signed Request** setting). For more information about request object, please refer to the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#RequestObject).

Parameter	Description
scope	The scope of the access request expressed as a list of space-separated, case-sensitive strings. Scope values are globally defined on the **OAuth Server# Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.
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 guidelines defined under *IETF BCP 47* (tools.ietf.org/html/bcp47).
idp (or PartnerIdpId)	A PingFederate OAuth AS parameter indicating the entity ID or the connection ID of the IdP with whom to initiate Browser 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 may 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.

If more than one source of authentication is configured in the system and no pfidpadapterid or idp parameter is provided, users are presented 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 is displayed 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 as a space-separated string, listing the values 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 is passed through unmodified from an authorization request to the ID token.

Parameter	**Description**
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, this parameter can be used by the client to verify that the end user is still present for the current session or to bring attention to the request. PingFederate supports the following values: `none`, `login`, `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 which would be 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 1: A partial match

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

Example 2: An exact match is a better match than a partial match

Access Token Management instances	**Resource URIs (configured)**

ATM1	`https://localhost:9031/app1`
	`https://localhost:9031/app2/data`
	`https://app.example.local`

Access Token Management instances	Resource URIs (configured)
ATM2	`https://localhost:9031/app1/data` `https://localhost:9031/app2/data/get`

`https://localhost:9031/app1` (a resource URI pre-configured for ATM1) is a partial match for https://localhost:9031/app1/data (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app1/data` (a resource URI pre-configured for ATM2) is an exact match against the provided URI.

Example 3: A more specific partial match is a better match

Both `https://localhost:9031/app2/data` (a resource URI for ATM1) and `https://localhost:9031/app2/data/get` (a resource URI for ATM2) are partial matches for https://localhost:9031/app2/data/get/sample (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app2/data/get` matches more specifically against the provided URI.

Client-initiated backchannel authentication endpoint

The client-initiated backchannel authentication endpoint is defined in the *OpenID Connect Client Initiated Backchannel Authentication Flow* specification (openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html). A CIBA-capable client uses this endpoint to initiate a backchannel, out-of-band flow to authenticate the resource owners and to obtain their authorizations.

Note: Per specification, 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. This is a required parameter. **Important:** When sending request parameters of an authentication request via a signed request object, the client must include the client_id parameter (and its value) inside and outside of the request parameter value. Furthermore, 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 **OAuth Server# Scope Management** screen. Scopes can also be constrained on a client-to-client basis. This is a required parameter. Additionally, it 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 in *RFC 6750* (tools.ietf.org/html/rfc6750#section-2.1). Required only if the client is configured to use the poll delivery method.
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. In other words, it must be a signed ID token. **login_hint_token** Use this parameter to include a 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 *Configure identity hint contract* on page 474.) **login_hint** Use this parameter to provide a hint to PingFederate to identify the user. The value may 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. Required only if the client record is configured to support user code and associated with a user code-enabled CIBA request policy.
binding_message	An alphanumeric message intended to be made available on both the authentication device and the consumption device so that the user can tie them together and decide whether to grant the requested authorization. When provided, the length of the message must range from 1 through 20.
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. Required if the client is configured to transmit request parameters to the backchannel authentication endpoint in signed request objects. (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. Optional 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 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 algorithm (or algorithms). If the signature does not pass the verification process, the request fails. **Important:** In the scenario where 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; for example:

```
HTTP/1.1 200 OK
...
{
    "auth_req_id":
 "HQnCgSeUzWNORZEv8n3E8wIip9L3iwBdJAAect04BqdpEsFBGqfxRvoa_Q",
    "interval": 3,
    "expires_in": 120
}
```

- 400 - Bad Request; for example:

```
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; for example:

```
HTTP/1.1 401 Unauthorized
...
{
    "error_description": "Invalid client or client credentials.",
    "error": "invalid_client"
}
```

- 500 - Server Error; for example:

```
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, refer to section *13. Authentication Error Response* in the specification (openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html#rfc.section.13).

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. Alternatively, clients can provide credentials using these request parameters: client_id and client_secret. **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body or through the use of the HTTP Basic authentication scheme, instead of in a query string.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject DN of the client certificate must match values configured for the clients.

Authentication method	Parameters

Private key JWT	Clients must include the 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 client authentication method, 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.
	For more information about the private_key_jwt client authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) and Using Assertions for Client Authentication in RFC7521 (tools.ietf.org/html/rfc7521#section-4.2).

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 which would be 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 1: A partial match

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

Example 2: An exact match is a better match than a partial match

Access Token Management instances	Resource URIs (configured)

ATM1	`https://localhost:9031/app1`
	`https://localhost:9031/app2/data`
	`https://app.example.local`

ATM2	`https://localhost:9031/app1/data` `https://localhost:9031/app2/data/get`

`https://localhost:9031/app1` (a resource URI pre-configured for ATM1) is a partial match for https://localhost:9031/app1/data (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app1/data` (a resource URI pre-configured for ATM2) is an exact match against the provided URI.

Example 3: A more specific partial match is a better match

Both `https://localhost:9031/app2/data` (a resource URI for ATM1) and `https://localhost:9031/app2/data/get` (a resource URI for ATM2) are partial matches for https://localhost:9031/app2/data/get/sample (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app2/data/get` matches more specifically against the provided URI.

Token endpoint

The token endpoint is defined in the *the OAuth 2.0 Authorization Framework* (tools.ietf.org/html/rfc6749#section-3.2) and used by the client to obtain an access token and possibly a refresh token by presenting its authorization grant. The token endpoint is used with every authorization grant except for the Implicit grant type (since an access token is issued directly from the authorization endpoint).

Note: Per specification, this endpoint accepts only the HTTP POST method.

Endpoint: /as/token.oauth2

Parameters vary depending on the grant type (see *OAuth grant type parameters* on page 792). 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 **OAuth Server**# **Authorization Settings** screen, the token endpoint is also accessible at such location.

For example, if the base URL is https://www.example.com:9031 and the **Token Endpoint Base URL** field value is https://www.example.local:9031, the token endpoints are accessible at the following locations:

- https://www.example.com:9031/as/token.oauth2, and
- 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. Alternatively, clients can provide credentials using these request parameters: client_id and client_secret. **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body or through the use of the HTTP Basic authentication scheme, instead of in a query string.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject DN of the client certificate must match values configured for the clients.

Authentication method	**Parameters**

Private key JWT	Clients must include the 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 client authentication method, 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.
	
	For more information about the private_key_jwt client authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) and Using Assertions for Client Authentication in RFC7521 (tools.ietf.org/html/rfc7521#section-4.2).

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 which would be 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 1: A partial match

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

Example 2: An exact match is a better match than a partial match

Access Token Management instances	Resource URIs (configured)

ATM1	`https://localhost:9031/app1`
	`https://localhost:9031/app2/data`
	`https://app.example.local`

ATM2	`https://localhost:9031/app1/data` `https://localhost:9031/app2/data/get`

`https://localhost:9031/app1` (a resource URI pre-configured for ATM1) is a partial match for https://localhost:9031/app1/data (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app1/data` (a resource URI pre-configured for ATM2) is an exact match against the provided URI.

Example 3: A more specific partial match is a better match

Both `https://localhost:9031/app2/data` (a resource URI for ATM1) and `https://localhost:9031/app2/data/get` (a resource URI for ATM2) are partial matches for https://localhost:9031/app2/data/get/sample (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app2/data/get` matches more specifically against the provided URI.

OAuth grant type parameters

Other parameters accepted by the `/as/token.oauth2` endpoint vary by the grant type being presented. They also include both OAuth-defined standard parameters and parameters proprietary to PingFederate. The grant type of the access token request is indicated by the following parameter:

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:pingidentity.com:oauth2:grant_type:validate_bearer` **Note:** Further parameters associated with each grant type are defined in the following sections.

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. Based on the code_challenge_method parameter value (if provided in the request for the authorization code in the first place), PingFederate OAuth AS validates the code_verifier parameter value against that of the code_challenge value. If the validation returns no error, PingFederate OAuth AS returns an access token (provided that there is no other error condition); 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 (tools.ietf.org/html/rfc7636), see *Authorization endpoint* on page 776.
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 (see *Authorization endpoint* on page 776). 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.

Parameter	Description
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. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

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:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
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. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

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:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

Parameter	Description
validator_id	A PingFederate OAuth AS parameter indicating the instance ID of the password credential validator to be used 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 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; for 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 detected by the LDAP Username PCV. OAuth-client developers may 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.

As needed, you may localize these messages by using the PingFederate localization framework for an international audience (see).

Note that the client_id parameter is not required when the **Allow unidentified clients to make resource owner password credentials grants** check box is selected on the **OAuth Server**# **Authorization Server Settings** screen.

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. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

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 reqest

```
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
```

Note that the auth_req_id parameter value 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* (tools.ietf.org/html/draft-ietf-oauth-device-flow) specification defines the process that allows a user to grant authorization to a device using a browser on a second device, such as a smartphone or a computer. For more information, see *Device authorization grant* on page 52.

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* (tools.ietf.org/html/rfc7523#section-2.1). **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scope values are globally defined on the **OAuth Server# Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

Note that the client_id parameter is not required when the **Allow unidentified clients to request extension grants** check box is selected on the **OAuth Server# Authorization Server Settings** screen.

SAML 2.0 Bearer Assertion grant type

These 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, as described in *RFC4648, section 5* (tools.ietf.org/html/rfc4648#section-5). ⓘ **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

Note that the client_id parameter is not required when the **Allow unidentified clients to request extension grants** check box is selected on the **OAuth Server**# **Authorization Server Settings** screen.

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:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.

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. The grant type allows an RS to check 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:** Alternatively, an RS client can 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; note that the value may 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"
}
```

Note that 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"
}
```

Note that 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 (see).

Once an ATM instance is chosen, PingFederate takes into consideration the per-instance session validation settings and processes the validation request (see).

Introspection endpoint

The introspection endpoint is defined in the *OAuth 2.0 Token Introspection* specification (tools.ietf.org/html/rfc7662) and used by a resource server (RS) client to validate an access token or a refresh token prior to granting access to a protected-resources call.

Note: Per specification, this endpoint accepts only the HTTP POST method.

Endpoint: /as/introspect.oauth2

The RS acts in the role of a client for the request/response exchange with the PingFederate OAuth AS to make the validation call using the following parameters. The required Content-Type value is `application/x-www-form-urlencoded` when transmitting via the HTTP POST method.

Parameter	Description
token (Required)	The bearer access token or refresh token to be validated. **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
token_type_hint	A hint about the type of token submitted for validation. PingFederate supports the following values: ▪ `access_token` ▪ `refresh_token` Required only when validating a refresh token.

Client authentication is not required. In other words, 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. Note that 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 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.
- 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.

Note: The response includes the username and subject (sub) elements only if they were mapped in the ATM instance.

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"
}
```

Note that 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"
}
```

Note that AAA1 and AAA2 are the expanded outcome of AAAGroup.

Response for a valid refresh token

```
{"active":true}
```

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. Alternatively, clients can provide credentials using these request parameters: client_id and client_secret. **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body or through the use of the HTTP Basic authentication scheme, instead of in a query string.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject DN of the client certificate must match values configured for the clients.

Authentication method	Parameters

Private key JWT	Clients must include the 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 client authentication method, 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.
	
	For more information about the private_key_jwt client authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) and Using Assertions for Client Authentication in RFC7521 (tools.ietf.org/html/rfc7521#section-4.2).

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 which would be 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 1: A partial match

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

Example 2: An exact match is a better match than a partial match

Access Token Management instances	Resource URIs (configured)

ATM1	`https://localhost:9031/app1`
	`https://localhost:9031/app2/data`
	`https://app.example.local`

ATM2	`https://localhost:9031/app1/data` `https://localhost:9031/app2/data/get`

`https://localhost:9031/app1` (a resource URI pre-configured for ATM1) is a partial match for https://localhost:9031/app1/data (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app1/data` (a resource URI pre-configured for ATM2) is an exact match against the provided URI.

Example 3: A more specific partial match is a better match

Both `https://localhost:9031/app2/data` (a resource URI for ATM1) and `https://localhost:9031/app2/data/get` (a resource URI for ATM2) are partial matches for https://localhost:9031/app2/data/get/sample (the provided URI). However, ATM2 is chosen because `https://localhost:9031/app2/data/get` matches more specifically against the provided URI.

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 (see).

Once an ATM instance is chosen, PingFederate takes into consideration the per-instance session validation settings and processes the validation request (see).

Token revocation endpoint

The token revocation endpoint is defined in the *OAuth 2.0 Token Revocation* specification (tools.ietf.org/html/rfc7009). It 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.

ⓘ **Note:** Per specification, this endpoint accepts only the HTTP POST method.

Endpoint: /as/revoke_token.oauth2

ⓘ **Important:** Direct access token revocation is only supported for *Internally Managed Reference Tokens*. Access tokens of the type *JSON Web Token (JWT)* 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.

If a refresh token is revoked, the associated access grant and access tokens will be revoked as well. If an access token is revoked, 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 **OAuth Server# Authorization Server Settings** screen, 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 via the HTTP POST method.

Parameter	Description
token (Required)	The token that the client wants to revoke. ⓘ **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body, instead of in a query string.
token_type_hint	A hint about the type of token submitted for revocation. PingFederate supports the following values: ▪ `access_token` ▪ `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. Alternatively, clients can provide credentials using these request parameters: client_id and client_secret. ⓘ **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body or through the use of the HTTP Basic authentication scheme, instead of in a query string.

Authentication method	Parameters
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject DN of the client certificate must match values configured for the clients.
Private key JWT	Clients must include the 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 client authentication method, 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. For more information about the private_key_jwt client authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) and Using Assertions for Client Authentication in RFC7521 (tools.ietf.org/html/rfc7521#section-4.2).
None	Clients must pass in the client_id parameter in a query string or the message body to identify themselves.

Grant-management endpoint

The grants endpoint (two are provided, one for use with parameters) is where resource owners go to view (and optionally revoke) the persistent access grants they have made. This endpoint is not part of the OAuth specification, but many OAuth providers offer a similar type of functionally. The grants displayed are those associated with the USER_KEY of the authenticated user. The same attribute mapping(s) from the authentication source to USER_KEY 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 shows 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 IdP with whom to initiate Browser 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 exactly the same way as the authorization endpoint.

Dynamic client registration endpoint

This runtime endpoint allows developers to register OAuth clients on PingFederate authorization server dynamically based the *OAuth 2.0 Dynamic Client Registration Protocol* specification (tools.ietf.org/html/rfc7591). In essence, developers can send client registrations with the desired properties (client metadata) to this endpoint. PingFederate evaluates the requests and returns a response with a client ID and the registered client metadata values if the requests are valid.

This runtime endpoint is only active when dynamic registration client is enabled and configured.

Important: Because dynamic client registration can expose your server to unwanted client registrations, it is recommended to protect 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 **OAuth Server# Client Settings** configuration wizard. 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: Per specification, this endpoint accepts only the HTTP POST method.

Endpoint: /as/clients.oauth2

Both the request and the response follow the specification.

Example 1

A developer wants to register a client that supports the authorization code flow, a couple 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
}
```

Note that PingFederate returns **201 Created**, the client ID, and other registered client metadata after creating the new client.

In addition, when a registration request does not specify a client authentication method (token_endpoint_auth_method), PingFederate defaults to client_secret_basic per RFC7591.

Example 2

A developer wants to register a client that supports the authorization code flow, refresh tokens, one redirection URI, one scope (profile), 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."
}
```

Note that 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, a couple 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.eyJzY29wZSI6WyJkQ1IiXSwiY2xpZW50X2lkX25hbW
CHtcQ79Wefz2Sw5GOB5LfV9mWJ0n3vzJ93Ie7wbEAkalIFg53J-9e7s59MjA1igx6ybflGMQ9QAjYob
jM24arJZZgopEXvcx6IQpyU8U4AMTJ7tr9Lmody8P0QZOKcUDBTT5egv9vr5NuXCtUBfVPhGZ-3p5g5
oDhmilbmiga4319YSFfX5-
U3li9XPeN3JZB2ukLbTFjjVIVLJIInbSR_IFTWP5Irg92aXLrIfm5MvBp8D1fOU6xYjbgjvw9QKNiFF
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
}
```

Note that 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"
}
```

Device authorization endpoint

The *OAuth 2.0 Device Authorization Grant* specification (tools.ietf.org/html/draft-ietf-oauth-device-flow) allows a user to grant authorization to a device client using a browser on a second device, such as a smartphone or a computer. 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 following table describes parameters for this endpoint. The required Content-Type value is `application/x-www-form-urlencoded`.

Parameter	Description
client_id	A unique identifier the client provides to the resource server (RS) to identify itself. This identifier is included with every request the client makes

Parameter	Description
scope (Optional)	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Scope values are globally defined on the **OAuth Server**# **Scope Management** screen. Scopes can also be constrained on a client-to-client basis. For detailed information about scopes, see *Scopes and scope management* on page 390.

Both the request and the response follow the specification.

Sample request

```
POST /as/device_authz.oauth2 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: www.example.com
...

client_id=df_client
```

Response codes and sample responses

- 200 - Success; for example:

```
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; for example:

```
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; for example:

```
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. Alternatively, clients can provide credentials using these request parameters: client_id and client_secret. **Important:** This is a sensitive parameter. To avoid recording it in web server logs, we recommend to only pass in this parameter (via the HTTP POST method) in the message body or through the use of the HTTP Basic authentication scheme, instead of in a query string.
Client certificate	Clients must present their client certificate for mutual TLS authentication. The issuer and the subject DN of the client certificate must match values configured for the clients.
Private key JWT	Clients must include the 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 client authentication method, 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. For more information about the private_key_jwt client authentication method, see Client Authentication in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication) and Using Assertions for Client Authentication in RFC7521 (tools.ietf.org/html/rfc7521#section-4.2).
None	Clients must pass in the client_id parameter in a query string or the message body to identify themselves.

User authorization endpoint

The *OAuth 2.0 Device Authorization Grant* specification (tools.ietf.org/html/draft-ietf-oauth-device-flow) allows a user to grant authorization to a device client using a browser on a second device, such as a smartphone or a computer. Based on specification, the user goes to PingFederate, the authorization server (AS), at its user authorization endpoint 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 via the HTTP POST method.

Parameter	Description
user_code (Optional)	The value represents the activation code.

Both the request and the response follow the specification.

Sample 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 to 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 confirmation 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; for example:

```
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"}
```

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. The configuration information returned by this endpoint is controlled by a template file and can be customized to suit multiple use cases simultaneously.

Note: This endpoint is only active when the OAuth AS role and the OpenID Connect protocol are enabled on the **System**# **Protocol Settings**# **Roles & Protocols** screen.

Note: This public endpoint accepts HTTP GET requests without authentication.

Endpoint: /.well-known/openid-configuration

The following table describes parameter for this endpoint.

Parameter	Description
policy_id	Indicates the OpenID Connect policy from which PingFederate derives the attributes to be included in the response body (under claims_supported). If omitted, PingFederate includes the attributes based on the default policy.

Sample 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,
    "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"
    ],
    "token_endpoint_auth_signing_alg_values_supported": [
        "RS256",
        "RS384",
        "RS512",
        "ES256",
        "ES384",
        "ES512",
        "PS256",
        "PS384",
        "PS512"
    ],
    "userinfo_endpoint": "https://localhost:9031/idp/userinfo.openid"
}
```

Some notable metadata parameters

CIBA user code support

The backchannel_user_code_parameter_supported parameter indicates whether the default CIBA request policy supports user codes, an optional feature in the CIBA specification.

In this 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* (openid.net/specs/openid-client-initiated-backchannel-authentication-core-1_0.html) and *Define a request policy* on page 473.

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 and , respectively.) Note that deploying PingFederate to run on a Java 11 environment will also have the same effect.

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, 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. As needed, 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 this example, because the **Token Endpoint Base URL** is set to `https://www.example.com:9031` on the **OAuth Server# Authorization Server Settings** screen, the token_endpoint parameter value is set to https://www.example.com:9031/as/token.oauth2 (see *Configure AS settings* on page 379 and *Token endpoint* on page 789).

UserInfo endpoint

OAuth clients can optionally present access tokens to the UserInfo endpoint for the purpose of retrieving additional information about their users, the resource owners. The amount of information is customizable through the use of one or more OpenID Connect policies. Information may include specification-defined attributes (standard attributes) and non-standard attributes. Scopes, authorized by the users, also determines the attributes to be returned.

Note: This endpoint is only active when the OAuth AS role and the OpenID Connect protocol are enabled on the **System# Protocol Settings# Roles & Protocols** screen.

Note: This endpoint accepts HTTP GET requests without parameter. Clients must present valid access tokens for authentication.

Endpoint: /idp/userinfo.openid

Example

```
$ 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"
}
```

Note that the self-contained access token in the Authorization HTTP header is truncated for readability.

If the access token presented is not valid, PingFederate returns to the client HTTP status code 401 Unauthorized.

Self-contained tokens

If clients using self-contained access tokens are expected to contact the UserInfo endpoint, care must be taken when configuring the **Client ID Claim Name** and **Scope Claim Name** settings in the Access Token Management (ATM) instance (or instances) that these clients use.

Client ID Claim Name

The default value of this field 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 removed and left blank, PingFederate does not include the client ID of the requesting client in the self-contained tokens. In this scenario, the ATM instance used by the default OpenID Connect policy must remain accessible to all clients (see *Define access control* on page 455); otherwise, clients using self-contained access tokens issued by this ATM instance (configured without a **Client ID Claim Name** field value) will not be able to retrieve additional claims from the UserInfo endpoint. Instead, they receive from PingFederate an HTTP status code 401 Unauthorized.

Scope Claim Name

The default value of this field 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 removed and left blank, PingFederate does not include any scope information in the self-contained token. As a result, clients using self-contained access tokens issued by this ATM instance (configured without a **Scope Claim Name** field value) will not be able to retrieve additional claims from the UserInfo endpoint. Instead, they receive from PingFederate an HTTP status code 403 Forbidden.

Web service interfaces and APIs

PingFederate provides two built-in, SOAP-accessible web services related to browser-based SSO. These services may be used by client applications to manage partner connections and support integration of web applications, respectively.

Connection Management Service

The Connection Management Service enables creation and deletion of single connection configurations in PingFederate. This service may 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 (see *Automating configuration migration* on page 251).

SSO Directory Service

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 may be required. For more information, see *Configure service authentication* on page 321.

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.

This feature is useful in a variety of circumstances, but the following primary use cases were considered:

- As a utility to migrate changes to a partner connection through staging environments (for example: development, test, production).

  Changes to URLs and keys may be needed to make the connection appropriate to the next environment.
- As a way for an external application to update or delete connections programmatically, or create new ones using an exported connection XML file as a template.

The WAR file for this service, `pf-mgmt-ws.war`, is located in the `<pf_install>/pingfederate/server/default/deploy2` directory.

Note: If you do not want to allow use of the service, it should not be deployed: 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 via a call to the Connection Management Service.

In either case, 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, data stores, and keypairs. Adapter instances and data stores are referenced by ID, and keypairs 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 *Access SP connections* on page 498 or *Access IdP connections* on page 595.

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, `IDP` or `SP`.

Code sample

The following example invoke 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 (data stores, keypairs, 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.

Once 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 via the PingFederate administrative console (see *Access SP connections* on page 498 or *Access IdP connections* on page 595). 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—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

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, `IDP` or `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. This allows a client of this web service to create or update a new connection (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

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. Therefore, because the XML is complex and validation is limited, attempting to create an XML connection from scratch is *not recommended*. Rather, the administrative console should be used to create the initial connection. 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 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.) This service allows web applications to avoid storing and maintaining the data locally. These types of data can be retrieved:

- A list of IdP partners

- A list of SP partners
- A list of IdP adapter instances
- A list of SP adapter instances

The SSO Directory Service provides information useful for integrating an application with a PingFederate server. It is a way for the application to find out dynamically which partners can be used for SSO. This means applications need not be modified when new partners are configured in PingFederate.

The WAR file for this module, `pf-ws.war`, is located in the `pingfederate/server/default/deploy` directory.

Note: If you do not want to allow use of the service, it should not be deployed: remove the WAR file from the `deploy` directory.

The service endpoint is:

```
pf-ws/services/SSODirectoryService
```

The WSDL document describing this service can be retrieved 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>` (see *SOAP request and response examples* on page 823).
- `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 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 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 (see *Configure service authentication* on page 321). We recommend the use of HTTPS 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. For example, the following is a typical SOAP request for an IdP list using the 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 will return 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. This service is provided primarily for organizations with many OAuth clients, allowing programmatic management of OAuth clients, as an alternative to using the administrative console, the administrative API, or dynamic client registration.

The *`/pf-ws/rest/oauth/clients`* and *`/pf-ws/rest/oauth/clients/<clientId>`* REST resources are URL path extensions of the PingFederate runtime endpoint; for example:

- 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 client records to be stored on an external storage.

ⓘ **Note:** Applications must authenticate to this web service using HTTP Basic authentication and credentials validated through an instance of a Password Credential Validator. The Password Credential Validator instance, in turn, must be selected in the OAuth AS configuration.

ⓘ **Tip:** The administrative API can also be used to 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 resource accepts the *POST*, *PUT*, and *GET* methods. The POST and PUT methods described in this section require parameter name/value pairs formatted in JavaScript Object Notation (JSON).

This resource accepts the POST, PUT, and GET methods. The POST and PUT methods described in this section require parameter name/value pairs formatted in JavaScript Object Notation (JSON).

POST

Use the POST method to creates a new client based on the parameters provided in the request. Parameters correspond to the fields on the **Client Management** screen. The required MIME type is `application/json`.

JSON Parameters

Parameter	Description
clientId (Required)	A unique identifier the client provides to the resource server (RS) to identify itself. This identifier is included with every request the client makes.
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 does not 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 (see the grantTypes parameter). ▪ This client signs its ID tokens using an HMAC signing algorithm (see the idTokenSigningAlgorithm parameter). ▪ This client is allowed to access the Session Revocation API endpoint (see the grantAccessSessionRevocationApi parameter). ▪ This clients sends a secret parameter value. ▪ Set to `SECRET` for HTTP Basic authentication. This authentication method requires the secret parameter. ▪ Set to `CLIENT_CERT` for mutual SSL/TLS authentication; recommended for client applications where security policies prohibit storing passwords. This authentication method requires the clientCertIssuerDn and clientCertSubjectDn parameters. ⓘ **Important:** If you choose mutual SSL/TLS authentication, you must configure a secondary PingFederate HTTPS port (see the property pf.secondary.https.port in the table under *Configuring PingFederate properties* on page 221). ▪ Set to `PRIVATE_KEY_JWT` check box if the client authenticates via the private_key_jwt client authentication method, as defined in *Client Authentication* in the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#ClientAuthentication).
secret	The client password or phrase. Required when the clientAuthnType parameter is set to `SECRET`.
clientCertIssuerDn	The issuer DN of the client certificate. Note that these are CA certificates imported into PingFederate on the **Security# Trusted CAs** screen. Alternatively, it may be set to `Trust Any` to trust all the issuers found on the **Trusted CAs** screen. 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 JWTs for client authentication. Applicable only when the clientAuthnType parameter is provided with a value of `PRIVATE_KEY_JWT`. Allowed 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 either a 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 and , respectively.) If this parameter is not 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 via the private_key_jwt client authentication method, to transmit request parameters using in signed request objects, or to do both. A valid value is either `true` or `false`. **Note:** The underlying Assertion Replay Prevention Service is cluster-aware (see).

Parameter	Description
requireSignedRequests	Indicates 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 OpenID Connect specification calls this JWT a *request object*. A valid value is either `true` or `false`. **Note:** If a client includes in an authorization request a request parameter (other than client_id and response_type) 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 and response_type request parameters, 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. It is also worth noting that if a request parameter is found only outside of the signed request object, such request parameter is dropped and ignored; no error message is returned. **Tip:** Per OAuth and OpenID Connect specifications, a client must always include in an authorization request the client_id, response_type, and scope request parameters outside of the signed request object. For more information about request object, please refer to the OpenID Connect specification (openid.net/specs/openid-connect-core-1_0.html#RequestObject). If this parameter is not provided, a value of `false` is assumed.

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 may send its authorization requests using request objects. Allowed 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 either a 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 and , respectively.) If this parameter is not 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. Only one of them is required if the client is configured to use the private_key_jwt client authentication method, to transmit request parameters in signed request objects, or to transmit CIBA request parameter in signed request objects (or to do any of them) so that PingFederate can verify the authenticity of the JWTs. In addition, either may also 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 runtime processing, see .) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be deployed to run in a Java 11 runtime environment or integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see .) Finally, if the client is configured to encrypt ID tokens using an asymmetric encryption algorithm (see the **ID Token Key Management Encryption Algorithm** setting), either the JWKS URL or the actual JWKS must be provided.
redirectUris	URIs to which the OAuth AS may redirect the resource owner's user agent after authorization is obtained. At least one redirection URI is required by the authorization code and implicit grant types.

Parameter	Description
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; 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 is either `true` or `false`. If this parameter is not provided, a value of `false` is assumed.
restrictScopes	This setting controls whether all existing common scopes and scope groups (and those created in the future) or only the select few should be made available to the client. A valid value is either `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 is not provided, a value of `false` is assumed. **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 can impact the results of scope evaluation. The default scope, however, is always allowed for and available to all clients. For detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section in *Scopes and scope management* on page 390.
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. The rest and any common scopes and scope groups created in the future become invalid for the client; that is, if the client tries to use such scope or scope group, it will receive 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. The rest and any exclusive scopes and scope groups created in the future become invalid for the client; that is, if the client tries to use such scope or scope group, it will receive an `invalid_scope` error message from PingFederate. If this parameter is not 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, however, is always allowed for and available to all clients. For detailed information, refer to the **Dynamic scope evaluation and per-client scope management** section in *Scopes and scope management* on page 390.
grantTypes	An array of one or more grant types, for which a client can request. Allowed 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) ▪ `urn:pingidentity.com:oauth2:grant_type:validate_bearer` (For more information about each grant type, see *Grant types*.)

Parameter	Description
restrictedResponseTypes	An array of one or more response types, for which a client can request. . Allowed values: ▪ `code` ▪ `code id_token` ▪ `code id_token token` ▪ `code token` ▪ `id_token` ▪ `id_token token` ▪ `token` For more information about these response types, see Definitions of Multiple-Valued Response Type Combinations (openid.net/specs/oauth-v2-multiple-response-types-1_0.html#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. Additionally, it is worth noting that the response types and grant types 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.

response type	grant types
`code`	`authorization_code`
`code id_token`	`authorization_code` and `implicit`
`code id_token token`	`authorization_code` and `implicit`
`code token`	`authorization_code` and `implicit`
`id_token`	`implicit`
`id_token token`	`implicit`
`token`	`implicit`

defaultAccessTokenManagerRef	The default Access Token Management (ATM) instance for this client.
validateUsingAllEligibleAtms	Applicable only to RS clients. If selected, this RS client is not required to specify additional parameters (access_token_manager_id or aud) to disambiguate the ATM instance in its token validation requests. When the RS client does not 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 check box is not selected by default.

Parameter	Description
requireProofKeyForCodeExchange	Applicable when the client is configured to support the `authorization_code` grant type. A valid value is either `true` or `false`. This field 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 (tools.ietf.org/html/rfc7636). When enabled, this client must include a one-time string value through the use of the code_challenge parameter in its authorization request (see). It must also submit the corresponding code verifier via the code_verifier parameter in its token request when exchanging an authorization code for an access token (see). If this paramter is not provided, the assumed value is `false`.
persistentGrantExpirationType	Overrides the **Persistent Grant Max Lifetime** field value set globally in the **OAuth Server**# **Authorization Server Settings** screen. Allowed values: ▪ `SERVER_DEFAULT` (the default): Use the global setting. ▪ `NONE`: Grants do not expire. ▪ `OVERRIDE_SERVER_DEFAULT`: Use in conjunction with the persistentGrantExpirationTime and persistentGrantExpirationTimeUnit parameters to set the expiration time period. **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 on the **OAuth Server**# **Authorization Server Settings** screen. Once added, the lifetime of persistent grants can be set 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, they can be configured 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`.

Parameter	**Description**
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`.
persistentGrantIdleTimeoutType	Overrides the **Persistent Grant Idle Timeout** field value set globally in the **OAuth Server**# **Authorization Server Settings** screen. Allowed values: ▪ `SERVER_DEFAULT` (the default): 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 (see). When an idle timeout value is configured without a maximum lifetime, persistent grants remain valid until they expire due to inactivity, or are revoked or removed. When an idle timeout value is configured with a maximum lifetime, persistent grants remain valid until they expire (due to inactivity or lifetime expiration) or are removed from the grant storage.
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`.

Parameter	Description
refreshRolling	Overrides the **Roll Refresh Token Values** setting configured globally in the **OAuth Server# Authorization Server Settings** screen. A valid value is either `true` or `false`. Note that a value of `true` does not override the **Minimum Interval to Roll Refresh Tokens (Hours)** value set on the Authorization Server Settings screen. If this parameter is not provided, the **Roll Refresh Token Values** setting configured globally setting in the **OAuth Server# Authorization Server Settings** screen is used.

OpenID Connect client settings

Note: The following parameters are only applicable when the **OpenID Connect** protocol is enabled in **System# Protocol Settings# Roles & Protocols** screen and this client supports the OpenID Connect use cases.

idTokenSigningAlgorithm	The JSON Web Signature (JWS) algorithm required for the OpenID Connect ID 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 either a 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 and , 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.

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
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 ▪ `AES-GCM-128` - A128GCM ▪ `AES-GCM-192` - A192GCM ▪ `AES-GCM-256` - A256GCM
policyGroupId	The desired Open ID Connect policy.
grantAccessSessionRevocationApi	Set to `true` to allow this client to access the Session Revocation API for back-channel session query and revocation. A valid value is either `true` or `false`. If this parameter is not provided, a value of `false` is assumed. **Note:** Generally speaking, if clients are allowed to add sessions to the revocation list, 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. (For more information, see .)

Note: If the **Track User Sessions for Logout** check box is selected in the **OAuth Server# Authorization Server Settings** screen, you can provide two additional parameters to enable asynchronous front-channel logout for this client.

Parameter	**Description**
pingAccessLogoutCapable	If set to `true`, PingFederate sends (via the browser) logout requests to an OpenID Connect endpoint in PingAccess as part of the logout process. A valid value is either `true` or `false`. If this parameter is not provided, a value of `false` is assumed.
logoutUris	A list of additional endpoints at the relying parties as needed. PingFederate sends (via the browser) requests to these URIs as part of the logout process. Note that the relying parties must return an image in their logout responses; otherwise, PingFederate returns an error message or redirect to the InErrorResource parameter value (if specified)..

Device Authorization Grant client settings

Parameter	Description
deviceFlowSettingType	This field controls whether to use global device authorization grant settings defined on the **OAuth Server# Authorization Server Settings** screen. Allowed 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, perhaps for ease of use or branding purposes, when formulating the verification URLs to be included in its device authorization responses (see *Device authorization endpoint* on page 808). 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 (see *User authorization endpoint* on page 811). Moreover, it 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*> **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

Parameter	Description
Client-initiated backchannel authentication (CIBA) client settings	
cibaTokenDeliveryMode	The token delivery method supported by the client. 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 CIBA grant type is enabled, this parameter is required; no value is assumed.
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	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`. If the CIBA grant type is enabled, this parameter is required; no value is assumed.
cibaPolicyId	The CIBA request policy associated with the client. PingFederate uses CIBA request policies to determine various aspects of CIBA authentication requests; for example, the maximum lifetime of authentication requests, the validity of unsigned login hint tokens, and the mapping configuration of identity hints. Provide an existing CIBA policy If this parameter is not provided and the CIBA grant type is enabled, PingFederate associates the client with the CIBA request policy has been designated as the default CIBA request policy on the **OAuth Server**# **Request Policies** screen.
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. 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 that when user code support is enabled, the associated CIBA request policy must also be user code-enabled.

Parameter	**Description**
cibaRequireSignedRequests	Indicates 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 OpenID Connect specification calls this JWT a *request object*. A valid value is either `true` or `false`. If this parameter is not provided and the CIBA grant type is enabled, CIBA signed requests are not required. Note that 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 ▪ `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 either a 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 and , respectively.) If this parameter is not provided and the CIBA grant type is enabled, the client can use any of the allowed signing algorithms.

Extended properties

Parameter	Description
extendedParameters	Provide values for extended client metadata fields; for example:

```
{
  ...,
  "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 multivalued client metadata field (ContactNumbers).

(Extended client metadata fields are defined on the **System# Extended Properties** screen.)

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 does not exist or is not authorized to create a client.
- 500 – Internal Server Error

  An unknown error has occurred.

PUT

Updates client details for a specified client.

Note: You cannot 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.

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.

A valid value is either `true` or `false`.

If this parameter is not provided, a value of `false` is assumed.

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 could not be updated.
- 401 – Invalid Credentials

  The user does not exist or is not 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 is not 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 could not be retrieved.
- 401 – Invalid Credentials

  The user does not exist or is not 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.

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 is not returned if the AS global setting is set for a client (the default).
- 400 – Failed To Retrieve Client

  The response contains details as to why client could not be retrieved.
- 401 – Invalid Credentials

  The user does not exist or is not 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 could not be deleted.
- 401 – Invalid Credentials

  The user does not exist or is not authorized.
- 405 – Method Not Allowed

  The client ID was not specified.
- 500 – Internal Server Error

  An unknown error has occurred.

Logging

PingFederate records the actions performed via this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you may 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.

OAuth Access Grant Management Service

PingFederate includes a REST-based web service to manage OAuth persistent grants administratively. This 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.

ⓘ **Note:** Applications must authenticate to this web service using HTTP Basic authentication and credentials validated through an instance of a Password Credential Validator. The Password Credential Validator instance, in turn, must be selected in the OAuth AS configuration.

Endpoints

Manage by client

`/pf-ws/rest/oauth/clients/<clientId>/grants[/<grantId>]`

ⓘ **Important:** Client records must be stored in an external client data store.

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) and DELETE (for revocation) methods. The results are formatted in JavaScript Object Notation (JSON).

Parameters

Parameter	Description
clientId	The client ID for which to retrieve or revoke grants. Required to manage grants for a specific client.
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 Mapping**, **Authentication Policy Contract Mapping**, or **Resource Owner Credentials Mapping** wizard.
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`. Note that 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 **OAuth Server# Authorization Server Settings** screen.

```
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* on page 108.)

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 via this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you may 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.

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 and the Access Token Management instance under which the access tokens issued can be used by one or more clients to manage persistent grants. Both settings are found on the **OAuth Server# Authorization Server Settings** screen. Additionally, clients, which are capable of managing grants on their users' behalves, 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 as a bearer token via the HTTP Bearer authentication scheme.

Tip: Not all OAuth use cases involve persistent grants. For more information, see *Transient grants and persistent grants* on page 108.

Endpoint: /pf-ws/rest/oauth/grants[/<*grantId*>]

This REST resource accepts the GET (for retrieval) and DELETE (for revocation) methods. The results are formatted in JavaScript Object Notation (JSON).

Parameter

Parameter	Description
grantId	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 by a previous GET request from this endpoint. **Note:** This parameter is required to revoke a specific grant.

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 scope (admin). The client then obtained an access token with such scope from PingFederate. The access token issued was a self-contained JWT (JSON Web Token).

This sample request illustrates a GET request from the client when it 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.)

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 via this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you may 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.

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. Note that this endpoint is not part of the OAuth specification. The **Grant Access to Session Revocation API** check box must also be selected 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.

Note: Generally speaking, if clients are allowed to add sessions to the revocation list, 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. (For more information, see .)

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.

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 JavaScript Object Notation (JSON).

Tip: Information about the Session Revocation API endpoint is also available in the OpenID Provider Configuration endpoint. Look for ping_revoked_sris_endpoint in the metadata.

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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUl
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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUl
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; for example:

```
GET /pf-ws/rest/sessionMgmt/revokedSris/
abc123&updateActivityTime=false
Host: localhost:9031
Authorization: Basic
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUl
X-XSRF-HEADER: PingFederate
```

Return codes

- 200 – OK

  `{"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

  `{"resultId":"session_mgmt_sri_not_revoked","message":"The SRI has not been revoked."}` if the session is not found in the revocation list. As previously stated, if PingFederate has been configured to manage authentication sessions and the request does not come with the `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 via this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you may 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.

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 *Access the API interactive documentation* on page 858. For known limitations, see *Release notes*.

Initial setup using the administrative API

After installing PingFederate, instead of using the administrative console to begin the initial setup process, you may make four unauthenticated administrative API requests to perform the following tasks:

1. A GET request to `/license/agreement` to retrieve an 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 administrative role **User Admin** (`USER_ADMINISTRATOR`) to the first local administrative account. Other administrative roles are optional at this point. For more information, refer to the interactive documentation for the administrative API (see *Access the API interactive documentation* on page 858).

Once the first local administrative account is created, you can make other authenticated administrative API requests to configure various components in PingFederate.

Authentication

Similar to the administrative console, access to the administrative API is protected after initial setup. The administrative API supports various authentication options, see *Configure access to the administrative API* for more information.

Concurrent access

The administrative API supports concurrent access. When concurrent API calls are made to modify the same API resource (such as the same IdP Adapter instance or the same SP connection), the last request processed by PingFederate wins.

Logging

PingFederate records actions performed via 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 is protected by one of the following authentication schemes:

- Native authentication (against local administrative accounts)
- LDAP authentication
- RADIUS authentication
- Mutual TLS client certificate-based authentication

For new installations, native authentication is chosen by default.

For upgrades, if the authentication method of the administrative API was not previously set (for example, 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 (for example, when upgrading from PingFederate 8.0 to a future release).

The authentication method for the administrative API can be changed at a later time to any of the four choices, regardless of which authentication method is chosen for the administrative console.

Besides authentication, 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	Admin	Configure partner connections and most system settings (except the management of local accounts and the handling of local keys and certificates).
Admin	Crypto Admin	Manage local keys and certificates.
Admin	User Admin	Create users, deactivate users, change or reset passwords, and install replacement license keys.
Auditor	*Not applicable*	View-only permissions for all administrative functions. When the **Auditor** role is assigned, no other administrative roles may be set.

Note: All three administrative roles are required to access and make changes through the following services:

- The `/ConfigArchive` administrative API endpoint
- The **System# Configuration Archive** screen in the administrative console
- The **Connection Management** configuration item on the **Security# Service Authentication** screen

Enable native authentication

About this task

When the administrative API is protected by native authentication, access to the administrative API is restricted to the users defined in the **Account Management** screen. 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. Verify the pf.admin.api.authentication value in `<pf_install>/pingfederate/bin/run.properties` is set to `native`.

   Update as needed and restart PingFederate to activate this change.

   > ⓘ **Note:** In a clustered PingFederate environment, you only need to modify `run.properties` on the console node.

2. Log on to the administrative console with an account that has the role User Admin.

   > ⓘ **Important:** When the administrative console is protected by an alternative console authentication (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 **System# Administrative Accounts** functionality unless the logged-on administrator has been granted the User Admin right.
   >
   > 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 such administrator logs on to the administrative console, the **System# Administrative Accounts** screen becomes available for her or him 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* on page 242.

3. On the **System# Administrative Accounts** screen, create or manage users as needed, including assigning various PingFederate administrative roles as indicated by the **PingFederate User Access Control** table in *Configure access to the administrative API* on page 854.

   > ⓘ **Note:** When assigning role(s), keep in mind that all users defined in the **System# Administrative Accounts** screen can be used to access the administrative API and the administrative console.

Enable LDAP authentication

About this task

When the administrative API is protected by LDAP authentication, the API calls must be authenticated by valid LDAP credentials over HTTP Basic authentication; otherwise, the administrative API returns an error message. The LDAP authentication setup, including role assignment, is available via `<pf_install>/pingfederate/bin/ldap.properties`. The roles assigned to the LDAP accounts affect the results of the API calls.

> ⓘ **Note:** When LDAP authentication is configured, PingFederate does not lock out accounts based upon the number of failed logon attempts. Responsibility for preventing access is instead delegated to the LDAP server and enforced according to its password lockout settings.

Steps

1. Verify the pf.admin.api.authentication value in `<pf_install>/pingfederate/bin/run.properties` is set to `LDAP`.

   Update as needed.

2. In the `<pf_install>/pingfederate/bin/ldap.properties` file, change property values as needed for your network configuration.

   See the comments in the file for instructions and additional information.

   Important: Be sure 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* on page 854.

   Note: When assigning role(s), keep in mind that all LDAP accounts specified in `ldap.properties` can be used to 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 via 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.

Enable RADIUS authentication

About this task

The RADIUS authentication setup is available via configuration files in the `<pf_install>/pingfederate/bin` directory. The RADIUS protocol provides a common approach for implementing strong authentication in a client-server configuration. The administrative API supports the protocol scenario for one-step authentication (for example, appending an OTP after the password).

When the administrative API is protected by RADIUS authentication, 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 RADIUS authentication is configured, PingFederate does not lock out accounts based upon the number of failed logon attempts. Responsibility for preventing access is instead 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. Verify the pf.admin.api.authentication value in `<pf_install>/pingfederate/bin/run.properties` is set to `RADIUS`.

   Update as needed.
2. In the `<pf_install>/pingfederate/bin/radius.properties` file, change property values as needed for your network configuration.

   See the comments in the file for instructions and additional information.

   Important: Be sure to 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 (also in the `bin` directory) to map LDAP group-based permissions to PingFederate roles. (For information about permissions attached to the PingFederate roles, see the **PingFederate User Access Control** table in *Configure access to the administrative API* on page 854.)

Note: When assigning role(s), keep in mind that all accounts specified in `radius.properties` can be used to 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.

Enable certificate-based authentication

About this task

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. In addition, the corresponding root CA certificate(s) must either be contained in the Java runtime or be imported into the PingFederate's Trusted CA store (see *Manage trusted certificate authorities* on page 292).

The rest of the certificate-based authentication setup, including specifying the Issuer DN of the root CA certificate(s) and the applicable role(s) of the client certificate(s), is available via `<pf_install>/pingfederate/bin/cert_auth.properties`. The roles assigned to the certificates affect the results of the API calls.

Steps

1. Log 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).

   To import a certificate, click **Trusted CAs** in the Certificate Management section under Server Configuration.

   Tip: You may wish to click the Serial number and copy the Issuer 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 (see the properties file for more information).

   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 role(s), 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* on page 854.

> ⓘ **Important:** The configuration values are case-sensitive.

> ⓘ **Note:** When assigning role(s), 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.

Access the API interactive documentation

About this task

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. In general, the API calls can be called 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.

Steps

- To access the administrative API documentation, follow these steps:
  a. Start PingFederate.
  b. Start a web browser.
  c. Browse to the following URL:

     https://<*pf_host*>:9999/pf-admin-api/api-docs/

     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 reachable from your computer.
- To test an administrative API, follow these steps:
  a. Select a section of the administrative API you would like to explore; for example, **/dataStores**.
  b. Expand the method you want to use; for example, **GET /dataStores**.
  c. Enter required parameters, if any.

     (For more information, see **Operation Models** underneath the selected API endpoint.)
  d. Click **Try it out**.

     > ⓘ **Note:** You may 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.

     If the request completes successfully, the administrative API returns the **Request URL**, the **Response Body**, the **Response Code**, and the **Response Headers**.

Attribute mapping expressions

PingFederate provides an advanced option allowing administrators to map user attributes by way of an expression language, Object-Graph Navigation Language (OGNL). Because the option carries with it a potential for misuse, however, it is disabled in the administrative console for security reasons.

CAUTION: The security concern posed by expressions is related to a potential for abuse by PingFederate administrative users within an organization; the concern is not related to any known external threats. We recommend, however, that the option be enabled only if required.

Enable and disable expressions

About this task

An administrator can manually enable or disable the use of expressions in PingFederate by editing a configuration file.

Important: If the current configuration contains expressions, disabling the feature causes errors during runtime processing.

Tip: When importing a configuration archive that uses expression mapping, the feature is enabled automatically.

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 evaluate Expressions to either `true` or `false` and save the file; for example:

```
<?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 a value (the installed default) *does not* necessarily disable the use of expressions. To facilitate backward compatibility, when no value is present, configuration archives containing expressions can be imported successfully, and further use of the feature is enabled. (The term “silent” is used for this condition in the server log.)

3. Start or restart PingFederate.

   Tip: If you are enabling expressions to use for mapping outbound provisioning attributes, it is not necessary to restart the PingFederate server.

4. If you have a clustered PingFederate environment:
   a. Sign on to the PingFederate administrative console.
   b. On the **System# Cluster Management** screen, click **Replicate Configuration**.
   c. Restart PingFederate on the engine nodes.

Results

When you enable expressions, these expressions are available for use in multiple locations:

- The **Source** list under each of the administrative-console contract fulfillment screens.
- The **Show Advanced Criteria** section on the **Issuance Criteria** screen following each of the administrative-console contract fulfillment screens.
- The provisioning attribute-mapping screen (when the **Outbound Provisioning** protocol is enabled).

Construct OGNL expressions

OGNL is based on the Java programming language. OGNL expressions are useful for evaluating and manipulating attribute values and returning information based on the results. You can also 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 IdP, PingFederate provides predefined OGNL variables for IdP-adapter attributes, any attributes retrieved from data stores, and attributes for token authorization. For an SP, variables are available for attributes received in an assertion, an attribute query, and attributes for token authorization. For example, the SAML_SUBJECT value may be retrieved using:

```
#SAML_SUBJECT
```

Note: Use the following construction for any attributes from any source that contain special characters (hyphens, for example), which 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:

```
#SAML_SUBJECT="usr@msn.com"
```

(Not `#SAML_SUBJECT=usr@msn.com`)

Data store syntax

For data-store attributes with an attribute source ID, use this syntax:

```
#this.get("ds.attr-source-id.attribute_name")
```

For data-store attributes without an attribute source ID, use this syntax:

```
#this.get("ds.attribute_name")
```

Issuance criteria syntax

To access mapped attributes when configuring token-authorization expressions, use this syntax:

```
#this.get("mapped.attribute_name")
```

To access most context attributes when configuring token-authorization expressions, use this syntax:

```
#this.get("context.attribute_name")
```

To access the HTTP Request context attribute, use this syntax:

```
#this.get("context.HttpRequest").getObjectValue()
```

Note: The returned value is an instance of `javax.servlet.http.HttpServletRequest` (see *docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletRequest.html*).

Sample OGNL expressions

Below are examples of using OGNL expressions for attribute mapping and token authorization.

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; for example:

```
<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 via a SAML SP connection. The latter is a JSON Web Token (JWT) via 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; statements calling methods whose arguments are enclosed in quotes must be entered 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::74c
```

The `isInRange` method supports both IPv4 and IPv6 CIDR notations.

HTTP request context

The following example may be used to retrieve a value from an HTTP request object. In this case, the expression retrieves the User-Agent HTTP header value and compare it against a value required for token authorization.

```
#this.get("context.HttpRequest").getObjectValue().getHeader("User-
Agent").equals("somevalue")
```

STS client authentication context

This STS SSL Client Certificate Chain example checks that the issuer of the client certificate matches the specified 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 *docs.oracle.com/javase/8/docs/api/java/security/cert/X509Certificate.html*.

Issuance criteria and multiple virtual server IDs

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 IdP) have their own departmental subdomains, sales.contoso.com and support.contoso.com. The SP identifies both environments under the parent domain, contoso.com.

In this scenario, the PingFederate IdP server can be configured 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. For example:

```
#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; statements calling methods whose arguments are enclosed in quotes must be entered on a single line.

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.c
```

Private key 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.o
#claims.get("aud")
```

Retrieving the entire private key JWT:

```
#this.get("context.HttpRequest").getObjectValue().getParameter("client_assertion")
```

Using the OGNL edit screen

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** screens or click **Test** in the **Show Advanced Criteria** section on the **Issuance Criteria** screen.

  Note: The test function does not work for the context.httpRequest attribute, because its value is an object rather then text.
- To test an expression:
  a. Enter an input value in the Value text box associated with the attribute.
  b. Click the **Test** link near the bottom right of the screen.

     If the expression contains no errors, the result appears under **Test Results**.

  Important: If you make changes to an expression and want to save it, click **Update** under **Actions**. To discard changes, click the **Cancel** *link* under **Actions**; clicking the **Cancel** button near the bottom of the screen discards all changes you made in the current task.

Customize assertions and authentication requests

About this task

Some Browser SSO use cases may require additional customizations in the assertions sent from the PingFederate IdP server to the SP or the authentication requests sent from the PingFederate SP server to the IdP. PingFederate is capable of fulfilling these use cases on a per-connection basis using OGNL expressions.

Steps

1. If you have not already done so, 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** screen, scroll down to the **Protocol Settings** section, and select the **URL** screen.
4. Click **Show Advanced Customizations** to begin customizing the applicable message.

   The available **Message Types** that can be customized varies depending your federation role (IdP or SP) as well as the protocol of the connection (SAML 1.x, SAML 2.0, and WS-Federation). Once a message type is selected, you have access to a list of the **Available Variables**. By calling various methods, you can customize the assertions or the authentication requests to fulfill your use case.

Message types and available variables

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 the `<pf_install>/pingfederate/sdk/doc` directory.

SP connections (SAML 2.0)

Message Types	Available Variables 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
ResponseDocument	#ResponseDocument org.sourceid.saml20.xmlbinding.protocol.ResponseDocument #Attributes org.sourceid.util.log.AttributeMap

SP connections (SAML 1.x)

Message Types	Available Variables 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 org.sourceid.protocol.samlp11.xml.ResponseDocument #Attributes org.sourceid.util.log.AttributeMap

SP connections (WS-Federation)

Message Types	Available Variables Classes/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 Variables Classes/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 Variables Classes/Interfaces in Javadoc
SP and IdP connections	SAML 2.0	#FedHubIncomingAuthnRequest org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument
SP connection	SAML 2.0	#FedHubOutgoingAuthnRequest org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument

Connections	**Protocol**	**Available Variables** **Classes/Interfaces in Javadoc**
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	#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.

Sample customizations

Below are examples of using OGNL expressions to customize assertions and authentication requests.

Add SessionNotOnOrAfter to assertions

This expression adds the optional SessionNotOnOrAfter attribute in the <AuthnStatement> element and set 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')?{
    #xml = #this.getAttributeValueArray(0).getStringValue(),
    #ast =
 @org.sourceid.saml20.xmlbinding.assertion.AttributeStatementType
$Factory@parse(#xml),
    #AssertionType.getAttributeStatementArray(0).setAttributeArray(#i,
 ast.getAttributeArray(0))
  }: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.

In this example, we use 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** screen. You are free to use other application logic here.

Sample inputs (attributes and their values)

Attribute Name	ExtAttr1
Attribute Name Format	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified
Attribute Value	`123`
Attribute Name	ExtAttr2

Attribute Name Format	urn:pingidentity.com:SAML:attrname-format:xml:complex
Attribute Value	`<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>`

```
<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>
```

Note: This is a well-formed XML document in one line.

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>
```

Fulfillment by data store queries

Data stores represent external systems where user attributes and other data are stored. Once defined, you can configure PingFederate to retrieve user attributes from data stores for contract fulfillment and token authorization in various use cases. PingFederate supports a wide variety of database servers and directory servers. As needed, you or developers at your organization can create custom drivers through the PingFederate SDK for connecting to other kind of data repositories, such as flat files or SOAP-connected databases. For more information, refer to 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.

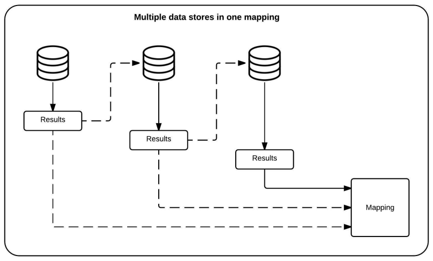
Tip: The Javadoc for PingFederate and the sample implementation are located under the `<pf_install>/pingfederate/sdk` directory.

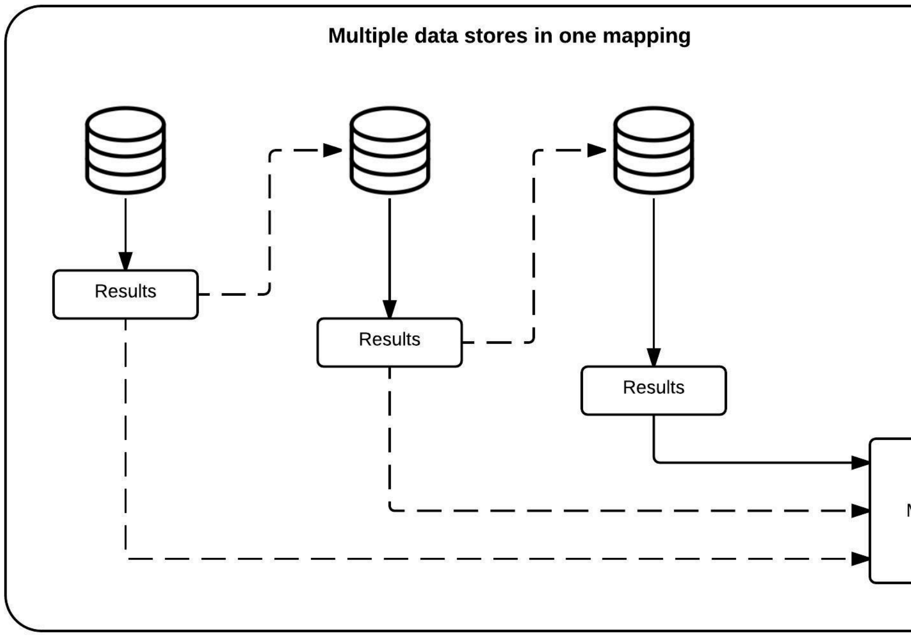
Attribute mapping with multiple data sources

You may configure PingFederate to query multiple data stores for additional attributes in most configurations.

Multiple data stores in one mapping

The PingFederate IdP server supports separate data stores to look up attributes for a single mapping. For example, you can query multiple data stores for values and map those values in one mapping, or query a data store for a value and use that value as input for subsequent queries of other data stores.





If a data store 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 data store in the setup. If you prefer PingFederate to abort and fail the requests, see Configure the behavior of searching multiple data stores with one mapping.

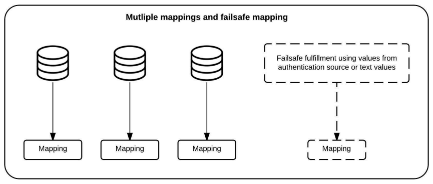
If a required attribute (such as SAML_SUBJECT in a SAML contact or subject in an authentication policy contract) cannot be fulfilled, the request fails.

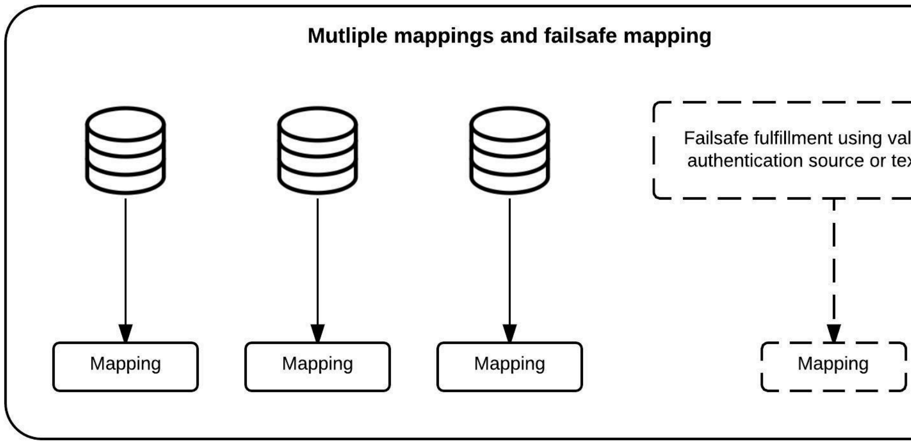
Multiple data stores in one mapping is available for Browser SSO and WS-Trust STS on the IdP side, authentication policy contract to SP adapter mappings, and the following OAuth workflows:

- 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 data store and for "fall-through" searches. For example, you can add the same data store more than once, using different search queries for each instance, or you can search different data stores successively. If any search fails to find a user in the specified attribute source, the next search is executed until a match is found. A failsafe attribute source can also be configured, 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


Mutliple mappings and failsafe mapping
Failsafe fulfillment using val
authentication source or tex
Mapping
Mapping
Mapping
Mapping

Data store query configuration

To query attributes from a data store, you must

- Choose a data store instance
- Specify the search location and the desired attributes
- Enter the search term to find the end users

Once the configuration is complete, you can fulfill a contract or verify a condition in the Token Authorization framework using the results from the data store queries.

Choose a data store

About this task

On the **Data Store** screen, choose a data store for PingFederate to look up attributes.

Steps

1. Enter a description (and ID if prompted) for the data store.
2. Select a data store instance from the **Active Data Store** list.

> **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.

3. Depending on the data store type, the rest of the setup varies as follows:

Data store type	Required tasks
JDBC	▪ ▪
LDAP	▪ ▪ (optional) ▪
Other	▪

Specify database table and columns

About this task

When you choose to retrieve attribute values from a database server, you follow this path through the configuration steps.

On the **Database Table and Columns** screen you begin to specify exactly where additional data can be found to fulfill your use case. Only one table may be used as a source of data for a database query. For more information about each field, refer to the following table:

Field	**Description**
Schema	Lists the table structure that stores information within a database. Some databases, such as Oracle, require selection of a specific schema for database queries. Other databases, such as MySQL, do not require selection of a schema.
Table	Displays the tables contained in the database. Select the table to retrieve data from the data store.
Columns to return from SELECT	Displays selected columns from the selected tables. Select the columns that are associated with the desired attributes you would like to return from the database queries.

Important: **(For MySQL users)** To allow for table and column names that may 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 JDBC connection string of your data store instance; for 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 *dev.mysql.com/doc/refman/5.7/en/server-system-variables.html*.

Steps

1. Choose a schema (when applicable) from the **Schema** list.
2. Select a table from the **Table** list.
3. Optional: Click **View Attribute Contract** to determine what attributes to look up.
4. Optional: Click **Refresh** if you are updating an existing configuration and changes may have been made to the database.
5. Under **Columns to return from SELECT**, choose a column name and click **Add Attribute**.

   Note: It is not required to add a column here for it to be used as part of a search filter. Add only the column that are required by subsequent sibling configuration items, such as contract fulfillment or token authorization. Any added columns that are left unused will be removed when the configuration is saved.

   Repeat this step to add more columns as needed.

Example

Suppose you (the IdP) have a data table named **ACCESSTABLE** with thee columns: userid, department, and accesslevel. Your use case requires accesslevel to be mapped into a SAML contract to an SP.

On the **Database Table and Columns** screen, select the **ACCESSTABLE** table and add the accesslevel column.

Enter a database search filter

About this task

On the **Database Filter** screen, enter a `WHERE` clause for PingFederate to query the database table you selected to retrieve a record associated with a particular value (or 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** screen.

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 IdP) or an assertion (for an 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 screen.

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. The initial `WHERE` is optional.
2. Ensure the syntax and variable names are correct. For more information about *WHERE* clauses, consult your DBMS documentation.
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 screen, the issuance criteria screen, or both, to fulfill your use case.

Example

Suppose you have selected a data table named **ACCESSTABLE** on the **Database Table and Columns** screen. 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** screen, 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.

Specify directory properties and attributes

About this task

When you choose to retrieve attribute values from a directory server, you follow this path through the configuration steps.

On the **LDAP Directory Search** screen you begin to 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 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.
Attributes	A list of attributes based on the selected **Root Object Class** value.

Steps

1. Optional: Specify a base DN.
2. Select a search scope.
3. Optional: Click **View Attribute Contract** to determine what attributes to look up.
4. Select a root object class, select an attribute, and then click **Add Attribute**.

   Note: It is not required to add an attribute here for it to be used 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 will be removed when the configuration is saved.

   Repeat this step to add more attributes as needed.

   Note: When connecting to 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, suppose 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.)

   For Oracle Directory Server, choose isMemberOf under **Attribute** for nested group membership. For more information, see *documentation about isMemberOf from Oracle* (docs.oracle.com/cd/E29127_01/doc.111170/e28967/ismemberof-5dsat.htm)

   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.

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** screen, 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**.

Define encoding for binary attributes

About this task

The **LDAP Binary Attribute Encoding Types** screen appears when at least one attribute is configured as such in the data store. Because binary attribute data cannot be used in an assertion to the SP, 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 IdP and IdP-to-SP bridging use cases.

Steps

To set an encoding type, select a value from the **Attribute Encoding Type** list.

Repeat this step for each binary attribute.

Examples

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.

Enter a directory search filter

About this task

On the **LDAP Filter** screen, enter a filter for PingFederate to query the data you selected to retrieve a record associated with a particular value (or values) from the user's session. The filter is in the form:

```
attribute1=value1
```

The left side (*attribute1*) is an attribute from your directory.

Tip: To get 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 IdP) or an assertion (for an 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 screen.

You can also apply additional search criteria by using other attributes from the target object class.

In general, 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) to search on, a search filter operator that will determine what to match, and the value of the attribute being sought.

Steps

1. Enter a search filter in the text field.
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 screen, the issuance criteria screen, 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** screen, 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.

Specify data source filter and fields

About this task

When you choose to retrieve attribute values from other types of data store, you follow this path through the configuration steps. Screens and steps depend on the implementation of the specific data store type.

Steps

- Refer to subsequent topics for configuration steps.

Specify a resource path for a REST API data store

About this task

Follow this path to set up attribute queries from a REST API data store.

Steps

On the **Configure Data Source Filters** screen, if the REST API data store requires a relative path or additional query parameters (or both) to retrieve user records, enter them in the **Resource Path** field.

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

Furthermore, 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 data store with a base URL of https://rest.example.com.
2. Add the REST API data store as an attribute source in the applicable use cases; for example, an SP connection that uses an HTML Form Adapter instance or an OAuth IdP Adapter Mapping configuration that maps from an HTML Form Adapter instance into the persistent grants. When prompted to configure the filtering option on the **Configure Data Source Filters** screen, enter `/staging/users?uid=${username}` in the **Resource Path** field.

Specify filters and fields for a custom data store

About this task

Follow this path to set up attribute queries from a custom data store.

Steps

1. On the **Configure Data Source Filters** screen you begin to specify a filter, or search query, for your data source. This screen display and the syntax of the filter depends on your developer's implementation of the PingFederate SDK.
2. On the **Configure Data Source Fields** screen, 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.

Results

Later in the workflow, you can use the values returned from your data source in the applicable contract fulfillment screen, the issuance criteria screen, or both.

Configure failsafe options

About this task

When a data store is configured and the attribute mappings under **Attribute Sources & User Lookup** fail to complete the attribute contract in an 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** screen does not appear if you have selected the **Retrieve additional attributes from multiple data stores using one mapping** option on the ***Mapping Method*** screen.

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 via the data-store mapping setup but others are not.

The failsafe mapping is used only when *all* of the mappings configured in the data-store 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** screen.
- To stop the SSO transaction, select **Abort the SSO transaction**.

Review data store query configuration

Steps

- To amend your configuration, click the corresponding screen title and 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 may also click **Save** as soon as the administrative console offers the opportunity to do so.
- To discard your changes, click **Cancel**.

Troubleshooting

Basic troubleshooting tips are provided here to help overcome common difficulties. Help is also available from the *Support Center*.

Enable debug messages and console logging

About this task

Starting with version 8.2, PingFederate only records messages that are tagged with log level `INFO`, `WARN`, `ERROR`, and `FATAL` to the server log (and the provisioner log). Messages that are tagged `DEBUG` (or `TRACE`) are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purpose, you may adjust the log level to `DEBUG` in the `log4j2.xml` file and optionally re-enable console logging.

Important: When debug messages and console logging are no longer required, ensure they are turned off.

Steps

1. Edit the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.
   a. To enable verbose messages, look for `Limit categories` inside the `Loggers` element, particularly the five logger names as illustrated in the following snippet.

```
...
<Loggers>

    <!-- ================ -->
    <!-- Limit categories -->
    <!-- ================ -->

    ...
    <Logger name="org.sourceid" level="INFO" />
    <Logger name="org.sourceid.saml20.util.SystemUtil" level="INFO"
 additivity="false">
        <AppenderRef ref="CONSOLE" />
        <AppenderRef ref="FILE" />
    </Logger>
    ...
    <Logger name="com.pingidentity" level="INFO" />
    <Logger name="com.pingidentity.common.util.ErrorHandler" level="INFO"
 additivity="false">
        <AppenderRef ref="CONSOLE" />
        <AppenderRef ref="FILE" />
    </Logger>
    ...
    <Logger name="com.pingidentity.appserver.jetty.PingFederateInit" level="ERROR"
 additivity="false" includeLocation="false">
        <AppenderRef ref="CONSOLE" />
    </Logger>
    ...
</Loggers>
...
```

Then, update five loggers as follows.

```
...
<Loggers>

    <!-- ================ -->
    <!-- Limit categories -->
    <!-- ================ -->

    ...
    <Logger name="org.sourceid" level="DEBUG" />
    <Logger name="org.sourceid.saml20.util.SystemUtil" level="DEBUG"
 additivity="false">
        <AppenderRef ref="CONSOLE" />
        <AppenderRef ref="FILE" />
    </Logger>
    ...
    <Logger name="com.pingidentity" level="DEBUG" />
    <Logger name="com.pingidentity.common.util.ErrorHandler"
 level="DEBUG" additivity="false">
        <AppenderRef ref="CONSOLE" />
        <AppenderRef ref="FILE" />
    </Logger>
    ...
    <Logger name="com.pingidentity.appserver.jetty.PingFederateInit"
 level="INFO" additivity="false" includeLocation="false">
        <AppenderRef ref="CONSOLE" />
        <AppenderRef ref="FILE" />
    </Logger>
    ...
</Loggers>
...
```

Note: In this snippet, console logging is enabled for loggers `org.sourceid.saml20.util.SystemUtil`, `com.pingidentity.common.util.ErrorHandler`, and `com.pingidentity.appserver.jetty.PingFederateInit`.

It is worth noting that console logging can be resource intensive. If you do not require console logging or prefer to review the server log instead, you can comment out the `<AppenderRef ref="CONSOLE" />` entry for these three loggers.

Tip: As needed, you may also update the log level for other loggers.

b. To enable console logging, look for the `Set up the Root logger` section; for example:

```
...
<!-- ======================= -->
<!-- Set up the Root logger  -->
<!-- ======================= -->
...
<AsyncRoot level="INFO" includeLocation="false">
    <!-- <AppenderRef ref="CONSOLE" /> -->
    <AppenderRef ref="FILE" />
</AsyncRoot>
```

Then, update as follows.

```
...
<!-- ======================= -->
<!-- Set up the Root logger  -->
<!-- ======================= -->
...
<AsyncRoot level="INFO" includeLocation="false">
    <AppenderRef ref="CONSOLE" />
    <AppenderRef ref="FILE" />
</AsyncRoot>
```

Important: When console logging is no longer required, comment out the `<AppenderRef ref="CONSOLE" />` entry for the `AsyncRoot` logger.

2. Save any changes made.

For a clustered PingFederate environment, repeat these steps on each applicable node.

Results

Changes, such as adding a `Logger` or adjusting log levels, are activated within half a minute. No restart of PingFederate is required.

Resolve startup issues

Problem	Solution
PingFederate does not start.	Make sure that a supported Java runtime is installed (see *Install Java* on page 68).
The server starts but indicates the license file is not found or invalid.	Ensure a current license is installed (see *Reviewing license information* on page 146).

Troubleshoot data store issues

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, username, and password. If unsuccessful, contact your database administrator. For more information, see *Configuring a JDBC connection* on page 167.
Cannot connect to a Directory Service with the LDAP protocol.	Verify the connection URL, port, principal, and credentials. If unsuccessful, contact your system administrator (see *Configuring an LDAP connection* on page 170). If using LDAPS, ensure the LDAP server's SSL certificate is signed by a trusted certificate authority or is imported into PingFederate (see *Manage trusted certificate authorities* on page 292).

Resolve URL-related errors

If a user encounters a `404 Not Found` status or a `System Error` message, check the URL of the request.

Examples

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* (www.w3schools.com/tags/ref_urlencode.asp).

The remedy for both sample issues is to update the URL of the IdP-initiated SSO to:

https://sso.idp.local/idp/startSSO.ping?PartnerSpId=sp1&TargetResource=https%3A%2F%2Fapp.sp1.local%2F

Resolve service-related errors

If a user encounters an `Unexpected System Error` message with a reference code, ask the user for the reference code and search the value in the server log. The log message should help determining the root cause, which usually requires a configuration change.

If a user encounters a *`partner`* `not active` status, select `Active` in the **Connection Status** field and click **Save** on the **Activation & Summary** screen for the connection.

Example

Unexpected System Error

When a PingFederate IdP server receives a SAML AuthnRequest message via the Redirect binding but such SAML profile is not selected in the applicable 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.

The remedy is to update the applicable SP connection to allow the Redirect binding for inbound messages from the SP if the Redirect binding is one of the mutually-agreed SAML bindings that both parties should use. Alternatively, the SP can send SAML AuthnRequest messages via an allowable SAML binding based on the configuration of your SP connection.

Troubleshoot authentication policy issues

About this task

Authentication policies, an optional configuration in PingFederate, help implement complex authentication requirements. Having a complex policy or multiple policies may result in unintended runtime behaviors. The `org.sourceid.util.log.PolicyTreeLogger` logger makes it easier to troubleshoot such issues.

Steps

1. Enable debug messages for the `org.sourceid.util.log.PolicyTreeLogger` class.

   Tip: If you have enable debug messages for the `org.sourceid` logger, you have already enable debug messages for the `org.sourceid.util.log.PolicyTreeLogger` class. Skip to step 3.

   a. Edit the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.
   b. Look for `Limit categories` inside the `Loggers` element, as illustrated in the following snippet.

```
...
<Loggers>

    <!-- ================ -->
    <!-- Limit categories -->
    <!-- ================ -->

    ...
    <!--
    <Logger name="org.sourceid.util.log.PolicyTreeLogger" level="DEBUG" />
    -->
    ...
</Loggers>
...
```

c. Uncomment the `org.sourceid.util.log.PolicyTreeLogger` logger; for example:

```
...
<Loggers>

    <!-- ================ -->
    <!-- Limit categories -->
    <!-- ================ -->

    ...

    <Logger name="org.sourceid.util.log.PolicyTreeLogger" level="DEBUG" />

    ...
</Loggers>
...
```

Note that the **<!--** and **-->** markers have been removed.

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. Once the issue is replicated, correlate server log messages using the PF cookie and tracking ID values (see *Troubleshoot runtime errors* on page 886).

   Look for `DEBUG` messages from the `org.sourceid.util.log.PolicyTreeLogger` class.

   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
```

(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.

a. 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.

b. 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.

c. Based on the user identifier provided by the user, PingFederate determines that the `Alpha` rule applies and routes this request to the `Alpha` policy path.

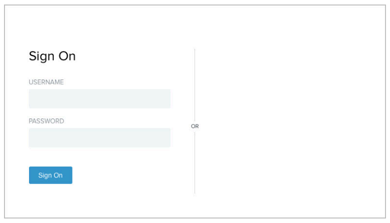
d. PingFederate routes this request to an IdP connection (`https://sso.alpha.local:8031`) because that IdP connection is the next authentication source of the `Alpha` policy path. Additionally, PingFederate populates the subject attribute in the AuthnRequest message with the user identifier obtained from the Identity First Adapter instance.

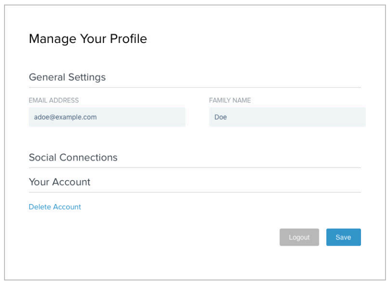
e. Eventually, PingFederate receives a valid security token from the IdP (`https://sso.alpha.local:8031`). PingFederate routes the request to the `Success` policy path, which ends with an authentication policy contract (`APC`) and concludes the authentication flow.

Troubleshoot registration and profile management issues

About this task

You have enabled third-party identity providers to provide users an alternative authentication option. Moreover, because you have enabled profile management, users can connect and disconnect their accounts at these 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.





Browser extensions, particularly those designed to block ads or social network components, may block the user interface elements representing the third-party identity providers you enabled.

Steps

- Verify whether users are using browser extensions that may have caused this issue. Disabling or fine-tuning the browser extension should resolve the issue.

Troubleshoot runtime errors

Users may encounter runtime errors when trying to connect to your partners. To troubleshoot these errors, investigating the user activities (in terms of the requests and the responses between the client and the PingFederate server) helps determining 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* (www.telerik.com/fiddler) and *Charles* (www.charlesproxy.com).

Server side

Review server log messages to investigate what PingFederate has received from the client.

On rare occasions, you may also use third-party tools, such as *Wireshark* (www.wireshark.org), or tools from the operating systems, such as `tcpdump` (www.tcpdump.org), to capture network traffic on the PingFederate server.

Tip: For instructions, please refer to the documentation of the third-party tools.

To correlate server log messages to user activities, you can use one of the following factors:

- The tracking ID, which can be configured to be shown in the user-facing error pages in PingFederate 7.2 (or a more recent version).
- 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.

Activate tracking ID in templates

About this task

You can configure PingFederate 7.2 (or a more recent version) to display the tracking ID in the user-facing error Velocity templates. When an error occurs, you can use the tracking ID to look for the related log messages. 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`

Note: You can find these Velocity template files in the `<pf_install>/pingfederate/server/default/conf/template` directory.

Steps

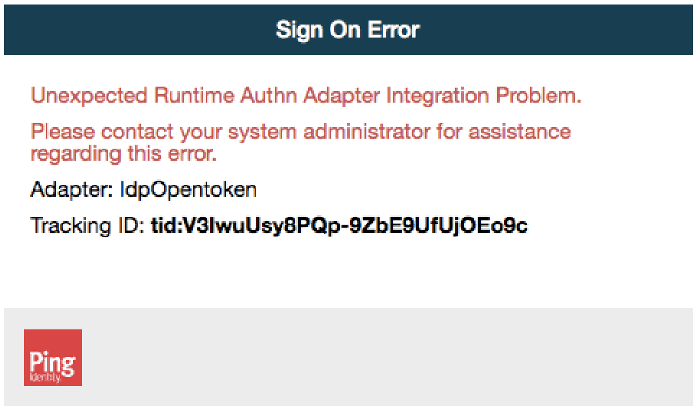
1. Open the applicable Velocity template file.

2. Search for the `$TrackingId` variable.
3. Follow the inline instructions to activate the variable.

   Template customization does not require a restart of PingFederate. For a clustered PingFederate environment, repeat these steps on each engine node.

Results

The following screenshot 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**

Correlate log messages by PF cookie

About this task

If you are using PingFederate 7.1 (or an earlier version), or 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.

Steps

1. Capture HTTP traffic and look for the PF cookie value.
2. Search for the PF cookie value in the server log.
3. As 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.

Results

Generally speaking, log messages that are tagged with `WARN` or `ERROR`, or prefixed with `Caused by` are most useful.

Example

Suppose an error had occurred and the associated the PF cookie value was `OaxBwPGw5OBeHVXe1sgifB7iZR5Rz2VI4rhJwqUSIXV`. 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 IdP adapter instance.

Correlate log messages by tracking ID

About this task

All server log messages (except the contents of the inbound requests and the outbound responses) are prefixed with their respective tracking IDs, which helps locating related log messages and payloads for a given transaction for troubleshooting.

Steps

1. Ask the user for the **Tracking ID** value in the error message.
2. Search for the tracking ID in the server log, for example:
3. Use the tracking ID to review log messages and payloads pertaining to this transaction.

Results

Generally speaking, log messages that are tagged with `WARN` or `ERROR`, or prefixed with `Caused by` are most useful.

Example

Suppose an error had 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.
```

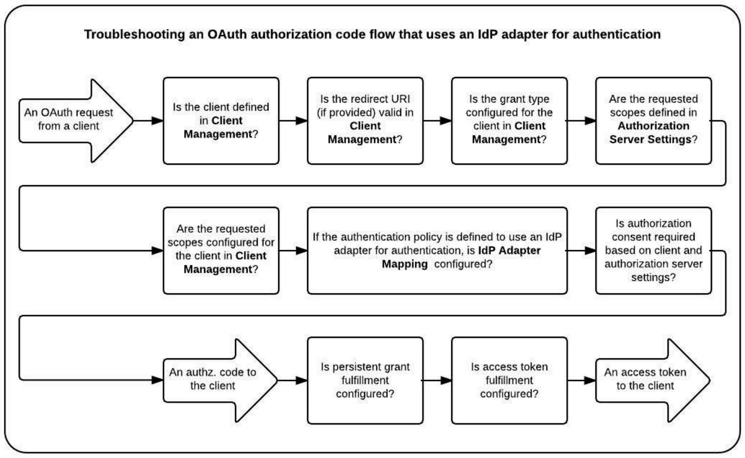
```
...
```

```
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 IdP adapter instance.

Troubleshoot OAuth transactions

Troubleshooting OAuth use cases involves reviewing the OAuth requests and various OAuth settings. This troubleshooting guide walks through an OAuth request by inspecting the parameters provided by the client in different stages of an OAuth transaction, and then compares the parameter values against the corresponding settings defined in the PingFederate administrative console.


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 IdP 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 may fail at which stage.

Review an OAuth request and various OAuth settings

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

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.
   a. Go to the **OAuth Server# Client Management** screen.
   b. Look for the client by its ID.

   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:

```
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.
   a. Go to the **OAuth Server# Client Management** screen.
   b. Select the applicable client.
   c. Compare the redirect_uri parameter value against the values defined in the **Redirect URIs** field.

   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:

```
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.
   a. Go to the **OAuth Server# Client Management** screen.
   b. Select the applicable client.
   c. Verify the `response_type` is selected in the **Allowed Grant Types** field.

   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:

```
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 (via the scope parameter) are defined for the authorization server.
   a. Go to the **OAuth Server**# **Authorization Server Settings** screen.
   b. Compare the scopes requested against the values defined in the **Scope Value** or the **Scope Group Value** fields.

   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:

```
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_scope#.
    SignaturePolicy: BINDING_DEFAULT
```

6. Check if the scopes requested are valid for the client.
   a. Go to the **OAuth Server**# **Client Management** screen.
   b. Select the applicable client.
   c. 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.

   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 IdP adapter for authentication. Check the IdP adapter mapping and the runtime selection made by the user.

   a. Go to the **OAuth Server**# **IdP Adapter Mapping** screen.
   b. Verify an entry exists for the IdP adapter involved.

   If more than one option is available, authentication policies may 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 also has the option to save the preferred authentication source for later (in the form of a `pfidpaid` cookie).

   If 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.
   a. Go to the **OAuth Server# Authorization Server Settings** screen.
   b. Review the **Reuse Existing Persistent Access Grants for Grant Types** setting.

      If the grant type is selected, the authorization consent page is bypassed for the same client, the same user and same (or lesser) scope.
   c. 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.
   d. Go to the **OAuth Server# Client Management** screen.
   e. Select the applicable client.
   f. Review the **Bypass Authorization Approval** setting.

      If the **Bypass Authorization Approval** check box is selected, the authorization consent page is bypassed as well.
9. 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** screen; this is the first stage of the two-stage OAuth attribute mapping process. Verify a mapping is configured.

   In this example, because the user authenticates via an IdP adapter, check the IdP adapter mapping.
   a. Go to the **OAuth Server# IdP Adapter Mappings** screen.
   b. Verify an entry exists for the IdP adapter involved and review its configuration.
10. 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 AS uses the following rules to determine which ATM instances it should use:

    a. Limit 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** screen. For OAuth Assertion Grant requests, it is the set of instances for which a mapping is defined in the IdP connection. Furthermore, the ACL (if configured) can also limits which ATM instances are eligible.

    b. If the request comes with an access_token_manager_id or aud parameter, PingFederate uses the information to determine the applicable ATM instance.

    c. 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 RS clients, you may optionally configure PingFederate to use any eligible ATM instances for the purpose of token validation.

    d. 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 RS clients do not provide either the access_token_manager_id or aud parameter in their requests and the RS clients have not been configured to validate against any eligible ATM instances, the same logic applies.

Review the request and the settings related to access token management.

a. 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.
b. Go to the **OAuth Server#** **Client Management** screen.
c. Select the applicable client.
d. Verify if a default access token is selected from the **Default Access Token Manager** list.
e. Go to the **OAuth Server#** **Access Token Mapping** screen.
f. Review the attribute mapping configuration for the authentication source (if such mapping exists) or the `Default` mapping.

Other runtime issues

Problem	Solution
Certificates unexpectedly expire.	Verify that the server clocks are synchronized on both sides of the federation. Note that you can configure PingFederate to notify administrators in advance of impending certificate expiration (see *Runtime notifications* on page 157).
Receive `CrossModule` or `Network` error messages when PingFederate is deployed with a supported HSM.	Verify network connections to the Hardware Security Modules (HSMs) are active and running. Also ensure the HSMs have not been unintentionally shut down.

Glossary

List of acronyms

ACS	Assertion Consumer Service
API	Application Programmer Interface
ARS	Artifact Resolution Service
CA	Certificate Authority
CRL	Certificate Revocation List
CSR	Certificate Signing Request
DBMS	Database Management System
DMZ	Demilitarized Zone
DN	Distinguished Name (certificate identifier)
DNS	Domain Name System
EIM	Enterprise Identity Management
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IdM	Identity Management

IdP	Identity Provider
IOT	Internet of Things
IP	Internet Protocol
JDBC	Java Database Connectivity (JDBC)
LDAP	Lightweight Directory Access Protocol
NAS	Network Access Server
O	Organization
OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
OU	Organizational Unit
PKI	Public Key Infrastructure
RADIUS	Remote Authentication Dial-in User Service
RDBMS	Relational Database Management System
RST	Request Security Token
RSTR	Request Security Token Response
SAML	Security Assertion Markup Language
SaaS	Software as a Service
SCIM	System for Cross-domain Identity Management
SDK	Software Development Kit
SP	Service Provider
SLO	Single Logout
SOA	service-oriented architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSL/TLS	Secure Sockets Layer/Transport Level Security
SSO	Single Sign-On
STS	Security Token Service
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WCF	Windows Communication Foundation
WIF	Windows Identity Foundation
WSC	Web Service Client
WSP	Web Service Provider
WSS	Web Services Security
XASP	X.509 Attribute Sharing Profile

XML	Extensible Markup Language

Server Clustering Guide

PingFederate provides clustering features that allow a group of PingFederate servers to appear to browsers and partner federation servers as a single system. In this configuration, all client traffic normally goes though a load balancer, which routes requests to the PingFederate servers in the cluster. User-session states and configuration data are shared among the servers, enabling them to process requests as a single entity.

Overview of clustering

Server clustering allows multiple PingFederate servers to share a single configuration and service single sign-on and logout requests as a single system.

When deployed appropriately, server clustering can facilitate high availability of critical services. Clustering can also increase performance and overall system throughput. It is important to understand, however, that availability and performance are often at opposite ends of the deployment spectrum. Thus, you (the administrator) may need to make some configuration tradeoffs that balance availability with performance to accommodate specific deployment goals. Some of these choices are identified throughout this topic.

Important: All servers in a cluster must use the same version of PingFederate.

Note: PingFederate provides separate failover capabilities specifically for Outbound Provisioning, which by itself does not require either load balancing or state management (see *Deploy provisioning failover*).

General architecture

The cluster architecture has two layers: the cluster-protocol layer and the runtime state-management services. In addition, these services can use different runtime state-management architectures when applicable.

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 leave and join the cluster.

Runtime state-management services

The runtime state-management services communicate session-state information required to process SSO and logout requests. PingFederate abstracts its runtime state-management services behind Java service interfaces. This enables PingFederate to use interface implementations without regard to underlying storage and sharing mechanisms. The abstraction also provides a well-defined point of extensibility in PingFederate. Depending on the chosen runtime state-management architecture, each service may share session-state information with a subset of nodes or all nodes.

Runtime state-management architectures

PingFederate supports adaptive clustering and directed clustering. Adaptive clustering offers the benefits of scaling PingFederate horizontally with no or barely any configuration requirement while directed clustering allows administrators to specify which runtime state-management service uses which architecture model.

Group-RPC oriented approach

The prepackaged state-management implementations are designed to accommodate a variety of deployments. The implementations leverage a remote-procedure-call (RPC) framework for reliable group communication, allowing PingFederate servers within a cluster to share state information.

Load balancing

Clustered deployments of PingFederate for single sign-on (SSO) and logout 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 via remote procedure calls as it processes SSO and logout requests.

PingFederate does not automatically balance the traffic among the servers in the cluster. SSO and logout requests must be managed externally to avoid overloading individual servers in the cluster. Because each server can only handle a certain amount of traffic, refer to *Performance Tuning Guide* to plan ahead.

The method that the balancer uses to select the appropriate server can vary from simple to highly complex, depending on deployment requirements. Specific balancing strategies, their strengths and weaknesses, as well as the impacts on PingFederate are discussed later (see *Runtime state-management architectures* on page 897).

Load balancers may incorporate SSL/TLS accelerators or work closely with them. Due to the high computational overhead of the SSL handshake, Ping Identity recommends terminating 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 (via the administrative console). A cluster may contain one or more runtime nodes but only one console node.

Note: The PingFederate administrative console node must be configured to run outside of the load-balanced group to successfully process SSO requests.

Cluster protocol architecture

PingFederate’s cluster-protocol services manage discovery, cluster messaging, connectivity failure detection, membership, and merging of split clusters. Nodes in the cluster *must* be able to communicate with one another over both the cluster bind port and the cluster failure detection port.

Important: 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

The static discovery method is suitable for a small cluster with about five to six engine nodes. Configuration requires no external component. Each node must be configured 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

The dynamic discovery method is well suited for environments where traffic volume may spike and require additional resources during the peak period to handle the increased traffic. Instead of configuring a static list of known nodes ahead of time, new nodes are configured to pull cluster membership information from a centralized repository. Because it is crucial that the information is safely stored and readily accessible by all nodes, 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, no coordination effort is required to maintain a static discovery list.

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 issues and updates the cluster with the new membership information. The mechanism does so by establishing TCP connections with other nodes at their cluster failure detection ports and sending network messages occasionally. When a node detects a failure, it propagates such condition to other nodes. As a result, the new cluster membership information is shared across the cluster.

Important: If any networking devices, such as a firewall, are deployed between nodes, they must be configured 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

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 may be desirable to share session-state information with only a subset of nodes for performance reasons. 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 are not required to modify individual configuration files to specify which nodes should participate in tracking user sessions.

In essence, each session is assigned an address from in an internally defined range. Each node is aware of which nodes are responsible for which segments of (session) addresses such that the entire range is covered. For redundancy, each session is stored on multiple nodes; 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.

Adaptive clustering is also designed to handle changes in cluster membership with ease. As individual nodes join and leave the cluster, adaptive clustering redistributes session-state information so that the replica set is maintained throughout the cluster.

The default size of a replica set is three, which provides redundancy in case two nodes in the replica set fail and ensures that requests are not delayed when a single node is slow to respond. The setting (replication.factor) is stored in the `cluster-adaptive.conf` file, located in the `<pf_install>/pingfederate/server/default/conf` directory.

Adaptive clustering is enabled for new installations; the pf.cluster.adaptive property in the `run.properties` file is set to `true`. For upgrades, if such property is not found or is set to `false`, adaptive clustering is disabled and directed clustering is used 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 located in the `<pf_install>/pingfederate/bin` directory.

Important: After making changes to the `cluster-adaptive.conf` and the `run.properties` files, you must apply the changes to all nodes in the cluster manually. The configuration replication process does not push these files across the cluster. Additionally, restart PingFederate if it is running.

Note: Adaptive clustering does not support the SAML 2.0 single logout (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).

Other advanced settings

As needed, you can 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 is not applicable to the Artifact-Message Persistence and Retrieval Service. Its messages are always shared across all nodes to fulfill their objectives. As needed, you can modify other applicable properties, such as the rpc.timeout property.

The following tables indicate the configuration file that applies to each implementation and the applicable properties. Refer to the indicated sections for detailed information about each implementation.

Configuration file and service implementation

Configuration file	RPC-based service implementation
cluster-account-locking.conf	
cluster-artifact.conf	
cluster-assertion-replay-prevention.conf	
cluster-idp-session-registry.conf	
cluster-inter-request-state.conf	

Configuration file	RPC-based service implementation
cluster-session-revocation.conf	
cluster-sp-session-registry.conf	

Property description

Property	Description
rpc.timeout	How long, in milliseconds, this node waits before timing out unresponsive RPC invocations. The default value is `500`, which is half a second.
synchronous.retrieve.majority.only	Indicates how many responses to wait for when making synchronous remote procedure calls (values: `true` or `false`). 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. Note this property is not applicable to the Account Locking Service and not found in the `cluster-account-locking.conf` file. The default value is `true`.
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 (possibly 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 whether PingFederate should process queries for revocation status by searching the local revocation list or collecting the information from other engine nodes in the cluster. 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). The default value is `true`.

Note: Other properties found in these configuration files, namely the preferred.node.indices and preferred.node.group.id properties, are ignored when adaptive clustering is enabled. (The latter is found only in the `cluster-idp-session-registry.conf` file.)

Multi-region support

When a cluster spans multiple regions, administrators may specify region identifiers for different groups of nodes. When regions are defined, PingFederate adjusts its algorithm such that 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 region and the remote regions. For requests that require read-only access to session-state information, the operations are performed locally for optimal performance. Furthermore, 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 automatically when region identifiers are configured (and adaptive clustering is enabled). Specifically, PingFederate provides cross-region support in the following areas:

- User session-state information maintained by the Inter-Request State-Management Service, the IdP Session Registry Service, and the SP Session Registry Service.

> **Note:**
>
> Per-SSO transaction states are not replicated cross-region. If, within the same SSO transaction, the user navigates or is redirected to different nodes in different regions or cluster node groups, the SSO transaction fails with the error `Unable to resume processing because saved state was not found for key`.
>
> To properly implement PingFederate adaptive clustering, it is expected that the user's browser would always return to the same region/cluster node group during the entire processing of an authentication policy. This would most commonly be done using a global load balancer that uses DNS to provide an IP appropriate to the user's geographic location. Network-level stickiness is not required within a given region/node group and is normally not recommended, because it can interfere with correct load balancing of application servers interacting with PingFederate.

- Assertion Replay Prevention Service.
- Account Locking Service.
- Replication, validation, and revocation of access tokens using the reference-token data model.

As needed, you can disable cross-region support in individual areas, in which case an engine node only pushes and pulls session-state information to and from the local replica set. To improve the accuracy of session-state information, you may 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.

OAuth access token management

When adaptive clustering is enabled, PingFederate shares reference token information with a replica set. 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 in the configuration file for adaptive clustering.

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 will not be able to de-reference, validate, or revoke reference-style access tokens that were 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.

Configure multi-region support

About this task

Steps

- To define a region identifier for a given node, update the node.group.id setting value in the `cluster-adaptive.conf` file.

  The `cluster-adaptive.conf` file, located in the `<pf_install>/pingfederate/server/default/conf` directory, is a per-server configuration.

  > **Note:** After making changes to the `cluster-adaptive.conf` file, restart PingFederate if it is running.

  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 setting value to `W` for the five nodes in the West Coast and `E` for the six nodes in the East Coast.

  Once defined, the identifiers for all nodes are displayed on the **System**# **Cluster Management** screen.
- To configure cross-region support for individual areas, follow the inline instructions in the `cluster-adaptive.conf` file to update the relevant setting values.

  (As mentioned earlier, configure each node as needed and restart PingFederate to activate changes made.)

Directed clustering

With directed clustering, administrators manually specify which PingFederate nodes should participate in tracking user sessions. Most of the group RPC-based service implementations make use of a *preferred-nodes* concept, which allows each node to have 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. Any changes must be replicated manually for each cluster node. The following tables indicate the configuration file that applies to each implementation and the applicable properties. Refer to the indicated sections for detailed information about each implementation.

> **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	
cluster-artifact.conf	
cluster-assertion-replay-prevention.conf	
cluster-idp-session-registry.conf	
cluster-inter-request-state.conf	
cluster-session-revocation.conf	
cluster-sp-session-registry.conf	

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 each subcluster. If specified, the group ID must be unique for each subcluster. 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. In addition, 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 in the **Identity Provider (or Service Provider)**# **Sessions** screen. 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 any 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`, which is half a second.
synchronous.retrieve.majority	Indicates how many responses to wait for when making synchronous remote procedure calls (values: `true` or `false`). 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`.
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 (possibly 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.

Property	Description
read.local.only (found only in the `cluster-session-revocation.conf` file)	Determines whether PingFederate should process queries for revocation status by searching the local revocation list or collecting the information from other engine nodes in the cluster. 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). The default value is `true`.

Preferred node indices

Configuring the preferred.node.indices property could reduce the memory footprint and network communications; however, care must be given as results vary depending on the volume of transactions and the distribution of them across engine nodes. Performance tuning could help in this regard. For more information, see *Performance Tuning Guide*.

Additionally, note that within a single cluster deployment, individual services can use different preferred nodes. In other words, you can set different values for the preferred.node.indices property for each service.

The use of preferred nodes can translate into any number of deployment configurations. Three primary strategies are discussed in the following sections and may serve as touch points for you to consider in conjunction with your network requirements:

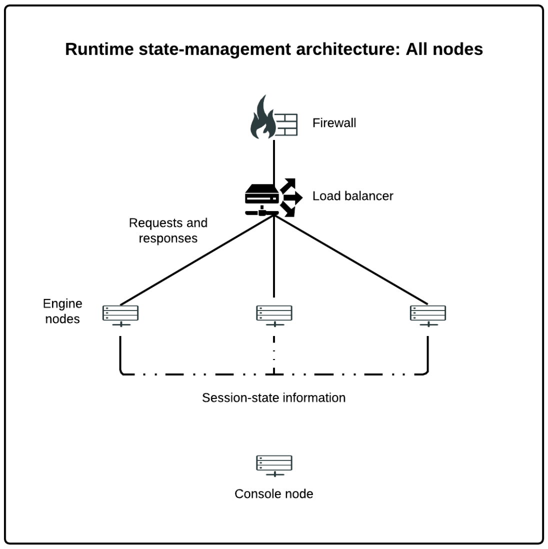
- *Sharing all nodes* on page 903
- *Designating state servers* on page 904
- *Defining subclusters* on page 905

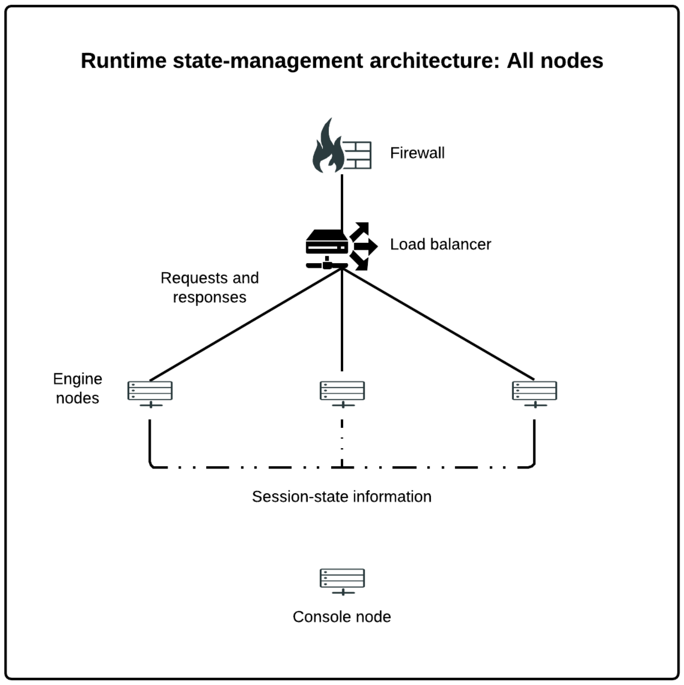
ⓘ **Note:** For PingFederate versions 7.0 and higher, it is possible to configure overrides for authentication-adapter processing based on the runtime node servicing a request (see *Configure the Cluster Node Authentication Selector*).

Sharing all nodes

Leaving the preferred.node.indices property blank in all the cluster-configuration files provides 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 directed to all nodes.


Runtime state-management architecture: All nodes
Firewall
Load balancer
Requests and responses
Engine nodes
Session-state information
Console node

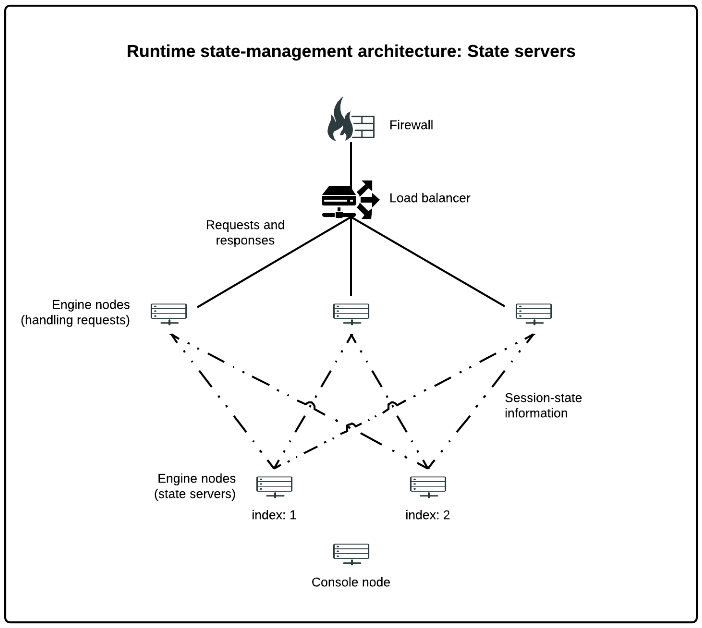

Runtime state-management architecture: All nodes
Firewall
Load balancer
Requests and responses
Engine nodes
Session-state information
Console node

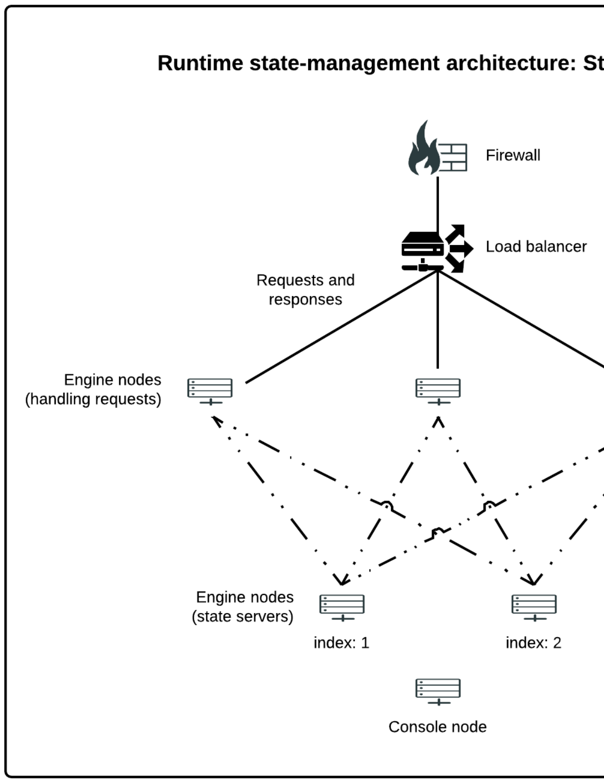
Designating state servers

You can select a few engine nodes as *state servers*. This deployment can be configured by setting the preferred-node indices of other servers in a group to those of the state servers. The load balancer should be configured to isolate the state-server nodes from end-user traffic. This approach scales better than the all-nodes approach because additional nodes do not require as much communication to every other node.

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


Runtime state-management architecture: St
Firewall
Load balancer
Requests and responses
Engine nodes (handling requests)
Engine nodes (state servers)
index: 1
index: 2
Console node

In this example, the two state-server nodes have indices of 1 and 2; therefore, the preferred.node.indices property of the engine nodes handling requests would be:

```
preferred.node.indices=1,2
```

And 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.

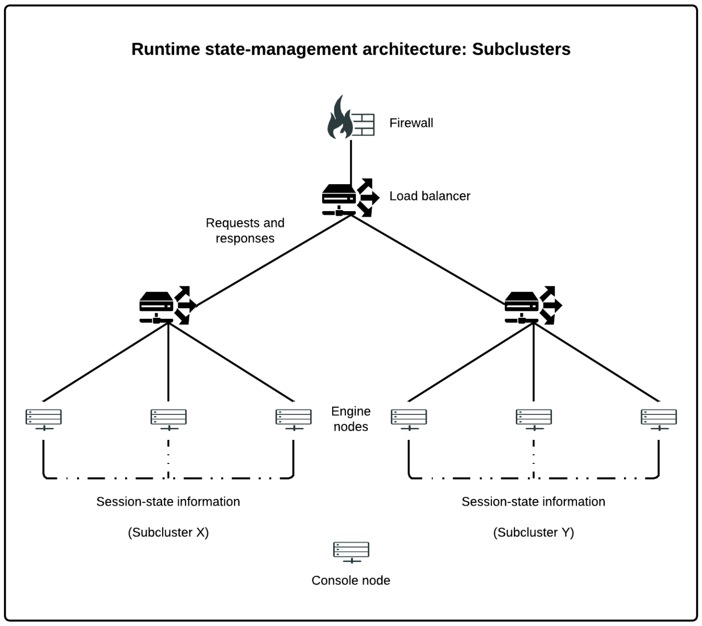
Defining subclusters

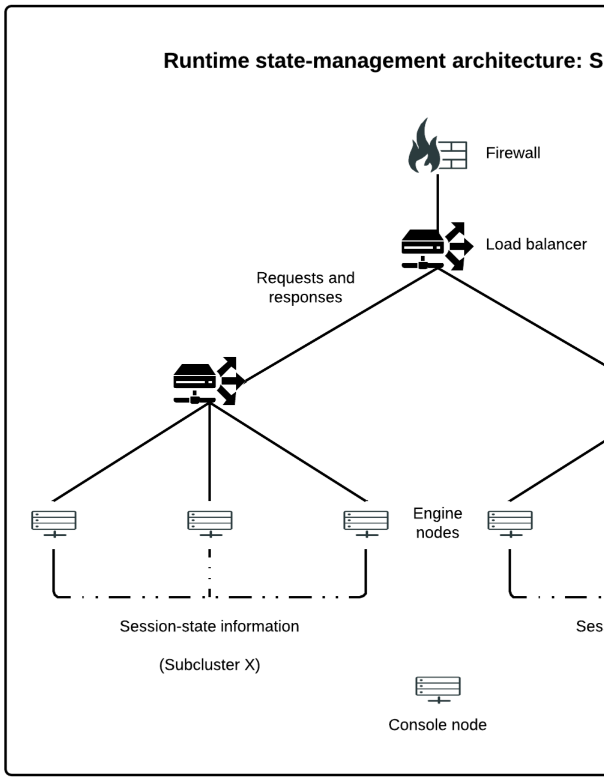
You can use node indices to divide a cluster into subgroups, or *subclusters*, of a few nodes each. Using this configuration, each node in a subcluster shares session-state information only with other members of the subcluster. This approach requires a network traffic management solution to persist, or *stick*, user sessions so that each subsequent request from the same user is directed to the same set of nodes. The advantage of this approach is that cluster throughput scales more linearly, because the creation of an additional subcluster will not degrade the performance of any other group.

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 the SAML 2.0 single logout (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.

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

The runtime state-management services are a collection of interfaces defining the contract that PingFederate uses with each service to manage session states. The implementation of each service interface is specified in the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF/` directory. This file does not need to be modified unless you wish to customize the way services are handled in a cluster. Any changes must be replicated manually for each cluster node.

By default, the interfaces listed in `hivemodule.xml` are proxies that select the best implementation for each service, based on the operational mode of the server. For example, if the server is in standalone mode, an in-memory-only implementation is used. If the server is in a clustered mode, a group RPC-based implementation is used. The proxies are provided for convenience; if you choose, you may specifically designate the implementation you desire for each service, as described in the following sections.

Configuration files for the services are located in the `<pf_install>/pingfederate/server/default/conf` directory.

Inter-Request State-Management (IRSM) Service

The PingFederate server tracks user-session state information between HTTP requests; for example, when PingFederate, acting as an 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 redirection. Another example is keeping track of state between issuing a request to a partner and processing the response. Generally, this state is short-lived.

The `InterRequestStateMgmtProxy` implementation chooses from among two methods available to track this state: group RPC-based (the clustering default) and local memory-based (the standalone default).

The configuration file is `cluster-inter-request-state.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

Group RPC-based session tracking

This 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 optionally override this default behavior in the configuration file for adaptive clustering. For directed clustering, all preferred-node approaches are possible with this implementation.

The service-point `InterRequestStateMgmt` in the `hivemodule.xml` file uses the proxy `InterRequestStateMgmtProxy` to use this implementation as the clustering default. The specific class name is:

```
org.sourceid.saml20.service.impl.grouprpc.InterRequestStateMgmtGroupRpcImpl
```

Local memory-based session tracking

In this alternative, the inter-request state of a user is tracked in the local memory of the processing server. This is the standalone default.

The service-point `InterRequestStateMgmt` in the `hivemodule.xml` file uses the proxy `InterRequestStateMgmtProxy` to use 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.

ⓘ **Important:** Adaptive clustering does not support this implementation. Use the group RPC-based session tracking instead.

IdP Session Registry Service

PingFederate uses the IdP Session Registry Service to facilitate single logout by tracking assertions issued to SP partners. This service is used only when the PingFederate server is acting in an IdP role and supports single logout 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 `cluster-idp-session-registry.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

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 adaptive clustering. 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 single logout (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).

The service proxy uses the class:

`org.sourceid.saml20.service.impl.grouprpc.IdpSessionRegistryGroupRpcImpl`

SP Session Registry Service

PingFederate uses the SP session registry service to facilitate single logout by tracking assertions issued from IdP partners. This service is used only when the PingFederate server is acting in an SP role and supports single logout 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 `cluster-sp-session-registry.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

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 adaptive clustering. 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 single logout (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).

The service proxy uses the class:

`org.sourceid.saml20.service.impl.grouprpc.SpSessionRegistryGroupRpcImpl`

LRU memory management schemes

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 as part of normal processing. However, it is common for end users to abandon web sessions, resulting in orphaned data.

To ensure that such 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. When a data structure reaches the maximum size, the oldest entries are automatically removed.

The maximum size of each data structure is configurable in the file `size-limits.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

Assertion Replay Prevention Service

The SAML standards specify that when an SP receives assertions via 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 optionally mandate a unique signed JWT from the client for each request when the client is configured to authenticate via the private_key_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 `cluster-assertion-replay-prevention.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

This 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, due to the nature of the threat that this service is intended to prevent, 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 many other security requirements are in place (for example, SSL, no-cache directives, and digital signatures), there may be situations where the priority placed on cluster performance outweighs the priority placed on this security check. If you are in this situation, you may wish to change the implementation for the service point `AssertionReplayPreventionService` in the `hivemodule.xml` 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; however, you may wish to use it, with caution, when performance is an absolute priority.

Artifact-Message Persistence and Retrieval Service

The following standards require PingFederate to relay data to partners via a reference-style data transportation model and to guarantee the reference keys are valid for one-time use only.

SAML artifact binding

When transmitting SAML outbound messages using the artifact binding, PingFederate sends to the partner an artifact. At a latter time, 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 to the client a code per 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.

Additionally, 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 keep tracks of the key and the associated data until the partner contacts the PingFederate server to exchange the key for the data.

When multiple PingFederate servers are deployed to form a cluster, the keys and their data are saved where they are generated. Because they are not replicated to other PingFederate servers, it is possible that a key resolution request can arrive at a server that does not hold the requested data. To handle this scenario, the Artifact-Message Persistence and Retrieval Service uses a group RPC retrieval approach, where the server handling the key resolution request determines the data-hosting server based on the key value and contacts such 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. The Artifact-Message Persistence and Retrieval Service also supports a local memory approach for SAML 2.0; however, it is only suitable in clustered environments where the OAuth 2.0 authorization code grant type and the Reference ID Adapter are not in -use.

Group RPC-based retrieval

When PingFederate is in clustered mode, the service proxy selects a group RPC-based implementation. This RPC implementation takes advantage of node indexing but not the preferred-nodes concept because sticky-session load-balancing strategies are not effective when the request that results in the issuance of a key 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, it does have a configuration file, `<pf_install>/pingfederate/server/default/conf/cluster-artifact.conf`, in which the RPC time-out can be configured.

SAML 2.0 indexing (local memory)

Due to the implementation complexity involved in managing state with the artifact binding, the SAML 2.0 specification introduced the concept of indexed endpoints. A SAML 2.0 federation entity may 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.

The downside to this approach is that partners must know about each of your back-end 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 zero in standalone mode).

Not only do partners need to know about each back-end server, they need direct access to each ARS endpoint. This may require more complicated configuration of load balancers, proxies, and firewalls. Another drawback to this approach is that it 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 `hivemodule.xml` file and change the implementation for the `ArtifactStore` 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 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. It does not use the preferred-nodes concept. This enables the flexibility of allowing the queries to be processed locally or results to be returned after collecting the information from other engine nodes; the former yields faster response time for engine nodes that are deployed in well-connected networks while the latter adds a layer of protection against inconsistent revocation lists among the engine nodes due to possible network outages.

The configuration file is `<pf_install>/pingfederate/server/default/conf/cluster-session-revocation.conf`. This is where the RPC time-out and other settings can be tuned.

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, in addition to the **Session Revocation Lifetime** setting (globally configured on the **OAuth Server# Authorization Server Settings** screen), 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 `<pf_install>/pingfederate/server/default/conf/size-limits.conf`. The default value is `50000`.

Account Locking Service

Account lockout protection prevents user accounts from becoming locked 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; for example, 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 locked out for a time period. 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 RPC-based implementation; the configuration file is `cluster-account-locking.conf`, located in the `<pf_install>/pingfederate/server/default/conf` directory.

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 optionally override this default behavior in the configuration file for adaptive clustering. 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.

Other services

Two other services may need consideration when running PingFederate in a cluster, depending on SAML 2.0 federation deployment needs:

- Account linking service
- Pseudonym service

Account linking service

This service stores the association between the external and internal identifiers of an end user when account linking is used as an SP identity-mapping strategy. The default, standalone implementation uses a JDBC interface to an embedded database within PingFederate. No information from the embedded database is shared across the cluster. Therefore, when account linking is used for an IdP connection deployed in a cluster, the default implementation will not work properly. In such cases, the pointer must be adjusted for cluster use by pointing the service to an external database (see *Define an account-linking data store*).

Pseudonym service

This service references the method needed by PingFederate to generate or look up a pseudonym for a user. The service is used 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. However, it may be desirable in some situations to implement pseudonym handling differently. Developers can refer to the Javadoc reference describing `PseudonymService` interface for more information.

Deploy cluster servers

About this task

Follow these steps to configure and deploy clustered PingFederate servers.

Note: Additional steps are required to set up failover for provisioning. Alternatively, if you are grouping servers *exclusively* to provide for provisioning failover, skip the following steps and refer to information under *Deploy provisioning failover*.

Steps

1. For each node in a cluster, install PingFederate.

2. For each node, edit its clustering properties in the `<pf_install>/pingfederate/bin/run.properties` file (see *Configure cluster protocol properties* on page 913).
3. Optional: For each engine node, edit configuration files that control the cluster protocol and runtime state-management services (see *Runtime state-management architectures* on page 897 and *Runtime state-management services* on page 906).
4. Optional: If outbound provisioning is configured at your site and you want to provide failover capabilities, identify and configure the provisioning failover nodes (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, as prompted (see *License management* on page 145).
8. On the **System**# **Cluster Management** screen, click **Replicate Configuration** to push the license information from the console node to all engine nodes.

   Note: Starting with PingFederate 8.2, you must use the **Replicate Configuration** screen 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.

Results

Once the clustered environment is set up, 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.

Configure cluster protocol properties

About this task

Follow these steps to configure the cluster protocol properties for all PingFederate nodes.

Steps

1. **1.** Edit the `run.properties` file, located in the `<pf_install>/pingfederate/bin` directory.

   Refer to the following table for information about each property.

Property	Description
pf.operational.mode	Controls the operational mode of the PingFederate server. PingFederate supports the following modes: **`STANDALONE`** This server is a standalone instance that runs both the administrative console and runtime engine. **`CLUSTERED_CONSOLE`** This server is part of a cluster and runs only the administration console. ⓘ **Important:** Only one node in a cluster can run the administrative console. **`CLUSTERED_ENGINE`** This server is part of a cluster and runs only the runtime engine. ⓘ **Important:** The value `STANDALONE` should not be used in a cluster unless session-state management is not needed for any reason and configuration-archive deployment is used as the configuration synchronization method. (The default value is `STANDALONE`.)
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. (Range: `0` to `65535`.) If no value is set for the node index, the system assigns an auto-generated value. (Range: `0` to `2147483647`.) ⓘ **Tip:** 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. (This property has no default value.)

Property	Description
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. (Value: any string, or blank.) Consider using a randomly-generated key with 22 or more alphanumeric characters as the property value. While optional, we recommend that you obfuscate the property value. For information about the **`obfuscate`** command-line utility, see its built-in help. All nodes in a cluster must share the same property value, blank or otherwise. (This property has no default value.)
pf.cluster.encrypt	Indicates whether or not to encrypt network traffic sent between nodes in a cluster. (Values: `true` or `false`.) 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 set for this property. (The default value is `false`.)
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`.)
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, search for the parameter bind_addr in *JGroups documentation* (jgroups.org/manual/index.html#Transport). **Important:** This field does not support DNS name. Use the default value (`NON_LOOPBACK`) or replace it with an IP address.

Property	**Description**
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`.)
pf.cluster.transport.protocol	Indicates the transport protocol used for cluster communication. (Values: `udp` or `tcp`.) Use UDP multicast when IP multicasting is enabled in the network environment and the majority of cluster traffic is point-to-full-group. In conjunction, you must 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 (routers discard multicast messaging). TCP may also be appropriate if your cluster configuration employs more point-to-point or point-to-few messaging than point-to-group. In conjunction, you must 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. As needed, you can customize either stack or add to it as needed by modifying the associated configuration file. All nodes in a cluster must have the same value set for this property. (The default value is `tcp`.)
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. (Range: `224.0.0.0` to `239.255.255.255`; note that 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. 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; 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 that is known in advance. In practice, this property should contain *all* nodes in the cluster (including itself) to increase the likelihood of new members finding and joining the cluster. Alternatively, leave this property blank and enable dynamic discovery in the `<pf_install>/pingfederate/server/default/conf/tcp.xml` file (see *Enable dynamic discovery for clustering* on page 917). (This property has no default value.)
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	Indicates if JGroups diagnostics is turned off (`false`) or on (`true`). (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.)

Note: This table does not include information about properties used for provisioning failover (see *Deploy provisioning failover*).

2. Repeat this process on the rest of the nodes in the cluster.

Important: You must manually configure the clustering properties on each node. The `run.properties` file is *not* copied from the console node to the engine nodes automatically; it is also *not* part of the **Replicate Configuration** process. PingFederate must be restarted if running.

Results

Tip: Other administrative-console and runtime behavior configuration options are also maintained in the `run.properties` file. For more information, see *Configuring PingFederate properties* on page 221.

Enable dynamic discovery for clustering

About this task

Dynamic discovery is well suited for environments where traffic volume may spike and require additional resources during the peak period to handle the increased traffic. This elastic scaling capability helps you to bring additional PingFederate engine nodes online with no additional configuration changes after the initial setup. PingFederate provides three dynamic discovery choices: `AWS_PING`, `S3_PING`, and `SWIFT_PING`.

With `AWS_PING`, you scale your PingFederate infrastructure using Amazon Elastic Compute Cloud (Amazon EC2) instances in the Amazon Web Service (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. It is worth noting that permissions to `ec2:Describe*` actions must either be enabled in the AWS Identity and Access Management (IAM) role assigned to the EC2 instance or be associated with the access_key that you provide as part of the dynamic discovery configuration. Furthermore, you may also use a combination of tags and filters, in which case only EC2 instances that satisfy both criteria are returned.

With `S3_PING` and `SWIFT_PING`, you have 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 reach out and find the new cluster. When PingFederate shuts down, it removes itself from the list and pushes an update to the repository.

Dynamic discovery configuration is maintained in the `tcp.xml` file, located in the `<pf_install>/pingfederate/server/default/conf` directory. No effort is required to maintain the pf.cluster.tcp.discovery.initial.hosts property in the `run.properties` file.

Important: You must manually configure or synchronize the dynamic discovery properties in the `tcp.xml` file on each node. The `tcp.xml` file is *not* synchronized automatically across the nodes in a cluster; it is also *not* part of the **Replicate Configuration** process. PingFederate servers must be restarted if running.

Steps

1. Configure cluster protocol properties.

   Edit the `run.properties` file, located in the `<pf_install>/pingfederate/bin` directory. Refer to the inline comments and the following table.

Property	Description
pf.operational.mode	Configure the operational mode of PingFederate. A a value of `CLUSTERED_CONSOLE` denotes a PingFederate administrative console node while a value of `CLUSTERED_ENGINE` denotes a PingFederate runtime engine node. In a PingFederate cluster, there is only one administrative console node. As 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.*	Refer to the inline comments to configure the rest of the pf.cluster.* property values.

   Important: You must manually configure the clustering properties on each node. The `run.properties` file is *not* copied from the console node to the engine nodes automatically; it is also *not* part of the **Replicate Configuration** process. If you require more information, see *Configure cluster protocol properties* on page 913.

2. Configure dynamic discovery properties.

Edit the `tcp.xml` file, located in the `<pf_install>/pingfederate/server/default/conf` directory. Refer to the inline comments and one of the following tables.

`AWS_PING`

Property	**Description**
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.
port_range	The number of additional ports that PingFederate may probe when attempting to connect to other nodes in the event that it fails to do so at the port specified by the port_number property. Suppose the port number is 7600, as determined by the port_number property, a port_range property value of `0` means PingFederate will only try to connect at port 7600; it will not try at a different port if it fails to connect at 7600. If the port_range property value is set to `2`, PingFederate can try up to two additional attempts; the port value increases 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`.
regions	A comma separated list of EC2 regions in which discovery should be attempted. If no regions are specified, only nodes in the same region as this node can be discovered. If nodes are running in multiple regions, it is recommended to list all regions in this property. For information about regions, see the *documentation* from Amazon (docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html).
tags	A comma separated list of EC2 tags. When specified, only EC2 instances that have been assigned with the specified tag (or tags) can be discovered. If multiple tags are specified, only EC2 instances that have been assigned with all tags can be discovered. For information about tags, see the *documentation* from Amazon (docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html).

Property	Description
filters	A semi-colon separated list of key value pairs of system metadata. When specified, only EC2 instances that match the specified filter (or 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 so long as 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. For information about filters, see the *documentation* from Amazon (docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html).
access_key and 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 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.
log_aws_error_messages	When set to `true` (the default), error messages received from AWS are logged to the server log.

`S3_PING`

Property	Description
location	The name of the bucket in your Amazon S3 environment. For information about buckets, see the *documentation* from Amazon (docs.aws.amazon.com/AmazonS3/latest/gsg/CreatingABucket.html).
access_key and secret_access_key	The security credentials to access your Amazon S3 environment. For information about both keys, see the *documentation* from Amazon (docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys). 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.
remove_all_files_on_view_change	When set to `true` (the default), JGroups cleans up files when it detects a view change. (For more information, see *jgroups.org/manual/index.html#FILE_PING*.)

Property	Description
write_data_on_find	The default value is `true`, which resolves the issue where subclusters could fail to merge after a network partition.

`SWIFT_PING`

Property	Description
auth_type	The authentication type.
auth_url	The authentication URL.
username and 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.
tenant	The name of your OpenStack Keystone tenant.
container	The name of the root container. (For more information about each of the `SWIFT_PING` properties, see *jgroups.org/manual/index.html#_swift_ping*.)
remove_all_files_on_view_change	When set to `true` (the default), JGroups cleans up files when it detects a view change. (For more information, see *jgroups.org/manual/index.html#FILE_PING*.)

3. Start or restart PingFederate.
4. Repeat these steps on the rest of the nodes (and any new nodes, as needed).

Results

After the initial setup, your nodes are ready to be deployed, undeployed, and redeployed as traffic volume changes.

It is worth mentioning that discovery mechanisms are separated from runtime state-management architectures. Discovery mechanisms determine how to find nodes to retrieve cluster information for the purpose of joining and rejoining such cluster. Runtime state-management architectures determine to and from which nodes session-state information is shared and fetched.

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* on page 897.

Note: Regardless of the chosen runtime state-management architecture, all nodes must still be able to communicate with other nodes for clustering-protocol messages. For additional information, see *Runtime state-management architectures* on page 897.

Deploy provisioning failover

About this task

After configuring outbound provisioning, you have the option to set up one or more failover PingFederate servers specifically for provisioning backup.

Provisioning runtime processing and failover is independent of SSO or 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.

> **Important:** The pre-installed HSQLDB database cannot be used in a failover configuration. Each server in the failover network must be configured to use the same relational database.

Steps

1. Identify 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`** 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. (The default value is `OFF`.)
provisioner.node.id	The unique index number of the provisioning server. Each server must have a unique index number (from `1` to *`n`*), which is used to prioritize which server is currently active and which is next in line in case of a failure. **Important:** The active primary server *must* have an index number of `1`. (The default value is `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**# **Protocol Settings**# **Outbound Provisioning** screen on the administrative console. (The default value is `600`, which is 10 minutes.)

Important: You must manually 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 **Internal Provisioning Data Store** setting in the **System**# **Protocol Settings**# **Outbound Provisioning** screen.

   Once configured, if the provisioning servers belong to the same PingFederate clustered environment, go to the **System**# **Cluster Management** screen and replicate the new **Internal Provisioning Data Store** setting to all nodes. If the provisioning servers are individual PingFederate servers, for each provisioning server, create a data store connection to the same external database and update the **Internal Provisioning Data Store** setting manually.

Configuration synchronization

All nodes in a PingFederate clustered environment must have the same configuration settings, as set via 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 (for example, in `<pf_install>/pingfederate/bin/run.properties`) must be replicated manually across the cluster, as applicable, and PingFederate servers must be restarted if running.

Console configuration push

When multiple PingFederate servers are set up to run as a cluster, the administrative console provides a **System**# **Cluster Management** screen.

Whenever applicable changes are made through the administrative console, a message appears at top of the console to serve as a reminder to go to the **Cluster Management** screen and to replicate the current console configuration to all server nodes in the cluster.

Note: Starting with PingFederate 8.2, you must also use the **Replicate Configuration** screen to initiate the transmission of the license file from the console node to all server nodes.

The **Cluster Management** screen is also useful for verifying the current member servers of your PingFederate cluster.

Configuration-archive deployment

After you configure or reconfigure the console, you can also update cluster nodes by downloading a configuration archive from the **System#** **Configuration Archive** screen and then deploying it (either manually or via a scripted process) to the `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory on each cluster node or provisioning-failover server.

A configuration archive contains the same information sent during the configuration push from the administrative console described in the previous section. However, this option 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 migrate the configuration files (located in the `<pf_install>/pingfederate/server/default/conf` directory) to the engine nodes manually.

Configuration file and service implementation

Configuration file	RPC-based service implementation
cluster-account-locking.conf	
cluster-artifact.conf	
cluster-assertion-replay-prevention.conf	
cluster-idp-session-registry.conf	
cluster-inter-request-state.conf	
cluster-session-revocation.conf	
cluster-sp-session-registry.conf	

SSO Integration Overview

Ping Identity® 's PingFederate must be integrated programmatically with end-user applications and identity management (IdM) systems to complete the "first- and last-mile" implementation of a federated-identity network. The purpose of this document is to provide an overview of the various approaches to integrating systems and applications with PingFederate for browser-based single sign-on (SSO). To enable both the Identity Provider (IdP) and Service Provider (SP) sides of this integration, PingFederate provides commercial integration kits, which include *adapters* that plug into the PingFederate server and *agents* that interface with local IdM systems or applications.

Integration introduction

As a standalone server, PingFederate must be integrated programmatically with end-user applications and identity management (IdM) systems to complete the “first- and last-mile” implementation of a federated-identity network. The purpose of this document is to provide an overview of the various approaches to integrating systems and applications with PingFederate for browser-based single sign-on (SSO). To enable both the Identity Provider (IdP) and Service Provider (SP) sides of this integration, PingFederate provides commercial integration kits, which include *adapters* that plug into the PingFederate server and *agents* that interface with local IdM systems or applications.

Integration kits, including various connectors for secure SSO to Software-as-a-Service (SaaS) providers, are available from our PingFederate Downloads website. User guides and other documentation for current integration kits can be found in the *PingFederate documentation* website.

PingFederate also includes a robust software development kit (SDK), which software developers can use to write their own custom interfaces for specific systems. Please refer to the PingFederate *SDK Developer's Guide* on page 931 for more information, available in the PingFederate distribution `sdk` directory.

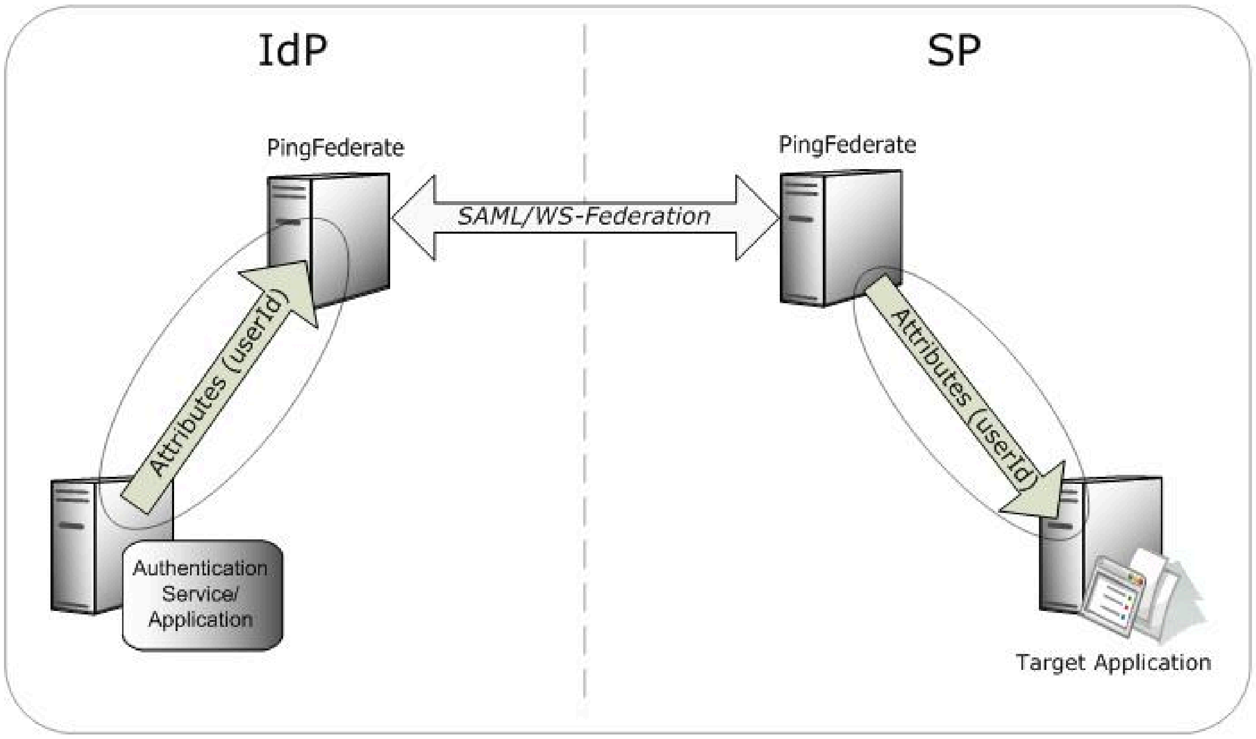
In addition, for integration with the PingFederate WS-Trust security token service (STS), we provide a range of *Token Translators*. These plug-in token processors (for an IdP) and token generators (for an SP) connect the STS with web service providers and clients for access to identity-enabled web services.

SSO integration concepts

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—Security Assertion Markup Language—refer to *Supported standards* on page 30.) IdP integration typically provides a mechanism through which PingFederate can look up a user's current authenticated session data (for example, 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.

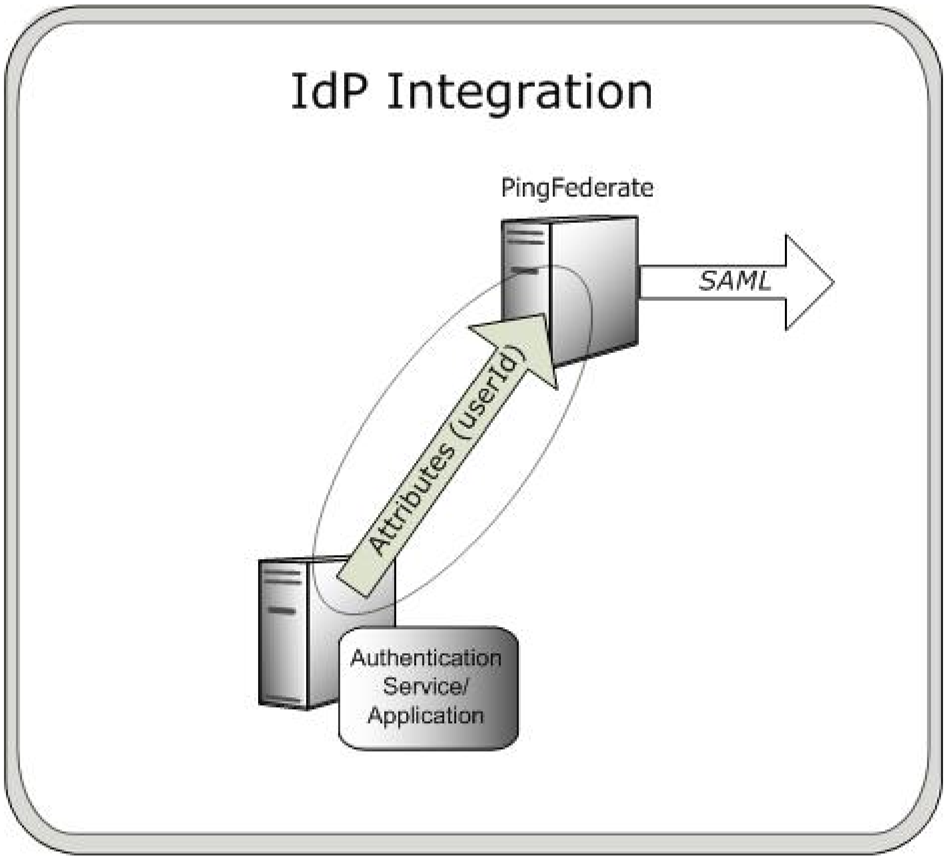
The following diagram illustrates the basic concepts of integration with PingFederate:



Identity provider integration

An IdP is a system entity that authenticates a user, or “SAML subject,” and transmits referential identity attributes based on that authentication to PingFederate. The IdP integration involves retrieving user-identity attributes from the IdP domain and sending them to the PingFederate server. Typically, the identity attributes are retrieved from an authenticated user session. For IdP integration, a number of attribute-retrieval approaches can be used, depending upon the IdP deployment/implementation environment. Ping Identity offers a broad range of commercial integration kits that address various IdP scenarios, most of which involve either custom-application integration, integration with a commercial IdM product, or integration with an authentication system.

Tip: For IdPs implementing SSO to selected Software-as-a-Service (SaaS) providers—for example, 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 through application-level integration kits, which 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. The information can be used 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.

Identity management (IdM) systems

An IdP enterprise that uses an IdM system can expand the reach of the IdM domain to external partner applications through integration with PingFederate. IdM integration kits typically use the IdM agent API (if available) 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; integration with PingFederate is accomplished entirely through the PingFederate administrative console.

Ping Identity provides integration kits for many of the leading IdM systems, such as Oracle Access Manager (formerly COREid).

Authentication systems

Initial user authentication is normally handled outside of the PingFederate server using an authentication application or service. PingFederate authentication-system integration kits leverage this local authentication to access applications outside the security domain. For example, an integration kit might access authenticated sessions that are validated against a Windows NTLM or Integrated Windows Authentication (IWA) environment, and then pass session attributes to the PingFederate IdP server.

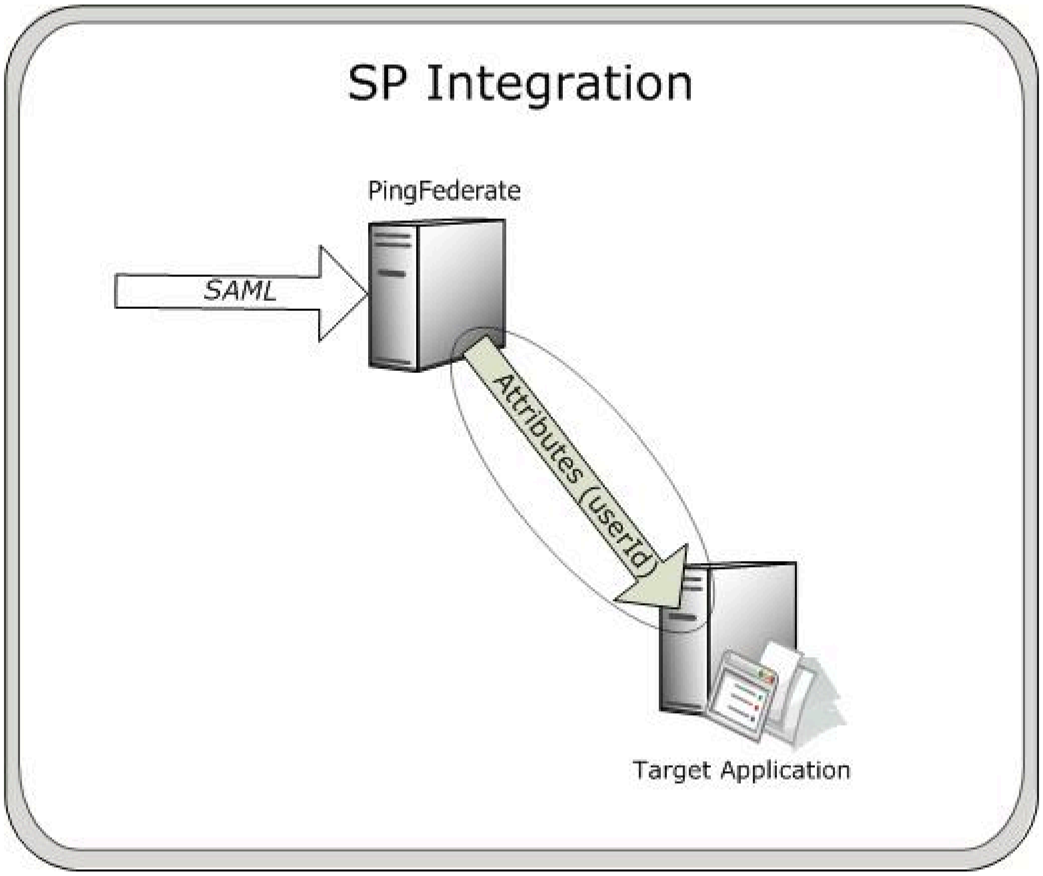
Authentication integration kits do not require any development; integration with PingFederate is accomplished entirely through the PingFederate administrative console. Ping Identity offers integration kits for authentication systems including:

- IWA/NTLM
- 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 plug-in password credential validators. Supplied validators can use either an LDAP directory, RADIUS server, or a simple username/password verification system maintained by PingFederate. (Customized validators may 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

An SP is the consumer of identity attributes provided by the IdP through a SAML assertion. SP integration involves passing the identity attributes from PingFederate to the target SP application. 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 can involve a number of approaches, and as 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 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. Integration with these custom applications is handled through application-level integration kits, which 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. The information can be used 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 SSO to all applications running on that web or application server; there is no need to integrate each application. Since integration occurs at the server level, ease of deployment and scalability are maximized.

Applications running on the Web/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 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-agent integration kits do not require any development; integration with PingFederate is accomplished entirely through the PingFederate administrative console.

Ping Identity provides integration kits for many web and application servers, including:

- Internet Information Services (IIS)
- Apache (Red Hat)
- Apache (Windows)
- NetWeaver
- WebSphere

Identity management (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 (if available) to create an IdM proprietary session token based on the identity attributes received from PingFederate.

IdM integration kits do not require any development; integration with PingFederate is accomplished through the PingFederate administrative console and the IdM administration tool.

Ping Identity provides integration kits for leading IdM systems, such as Oracle Access Manager (formerly COREid).

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 do not require any development; integration with PingFederate is accomplished entirely through the PingFederate administrative console.

Ping Identity offers integration kits to many commercial applications and SaaS vendors, including:

- Citrix
- SharePoint
- Box
- Google
- Office 365
- Salesforce.com
- Slack
- Workday
- Zendesk

Summary

The following table summarizes IdP- and SP-integration deployment scenarios with bundled adapters and some of the integration kits that suit each scenario.

Deployment scenarios	IdP	SP
Custom Application	.NET Integration Kit Agentless Integration Kit Java Integration Kit PHP Integration Kit	.NET Integration Kit Agentless Integration Kit Java Integration Kit PHP Integration Kit
Identity Management System (IdM)	Web Access Management (WAM) Integration Kit	WAM Integration Kit
Authentication System	Kerberos Adapter (bundled with PingFederate) HTML Form Adapter (bundled with PingFederate) HTTP Basic Adapter (bundled with PingFederate) Windows IWA Integration Kit X.509 Integration Kit RSA SecurID Integration Kit Symantec VIP Integration Kit	Not applicable
Server Agent	NetWeaver Integration Kit	Apache Integration Kit (Red Hat) Apache Integration Kit (Windows) IIS Integration Kit NetWeaver Integration Kit WebSphere Integration Kit
Software as a Service (SaaS)	Not applicable	AWS Connector Box Connector Concur Connector Dropbox Connector Evernote Connector Google Connector Office 365 Connector Salesforce Connector ServiceNow Connector Slack Connector Workday Connector Zendesk Connector

Ping Identity continues to develop new bundled adapters (included with PingFederate) and integration kits; check the PingFederate Downloads website for the most up-to-date list of kits.

User guides and other documentation for current integration kits can be found in the *PingFederate documentation* website.

SDK Developer's Guide

The PingFederate SDK enables integration with IdPs and/or SPs. The interfaces allow developers to build their own custom implementations for communicating authentication and security information between PingFederate and the enterprise environment.

Preface

This document provides technical guidance for using the Java Software Development Kit (SDK) for PingFederate. Developers can use this *Guide*, in conjunction with the installed Javadocs, to extend the functionality of the PingFederate server.

Intended Audience

The *Guide* is intended for application developers and system administrators responsible for extending PingFederate, including development of:

- Authentication adapters needed to integrate web applications or identity-management systems (when not already available: see the PingFederate *SSO Integration Overview on page 924* , described under *Additional Documentation* on page 931.)
- Authentication selectors used to direct SSO authentication to instances of authentication adapters based on specified conditions
- WS-Trust Security Token Service (STS) token translators, including token processors needed to consume and validate security tokens and token generators needed to create security tokens
- Custom data source drivers
- Password credential validators
- Identity store provisioners

The reader should be familiar with Java software-development principles and practices.

Additional Documentation

- Javadocs provide detailed reference information for developers. The Javadocs are located in the `<pf_install>/pingfederate/sdk/doc` directory.
- The PingFederate *SSO Integration Overview on page 924* describes the types of prebuilt authentication adapters Ping Identity provides for integrating web applications and identity-management systems with PingFederate. Since these adapters are based on the SDK, you may want to review this document before building your own adapter to see if your needs have already been met.
- The PingFederate *Administrator's Manual on page 99* provides background information and user-interface (UI) configuration details needed to integrate implementation(s) of PingFederate interfaces.
- The *user guides for the Java, .NET, and PHP integration kits* show examples of SDK implementations.

SDK introduction

The PingFederate Java SDK consists of several application programming interfaces (APIs), including:

- Adapter and STS Token-Translator interfaces
- Authentication selector interfaces
- Custom data source interfaces
- Password Credential Validator interfaces

- Identity Store Provisioner interfaces

Each of these interfaces allows users to create their own plug-ins, customizing certain behaviors of PingFederate to suit an organization's needs. This SDK provides a means to develop, compile, and deploy custom plug-ins to PingFederate.

A number of example plug-ins are included in the PingFederate package for reference. The example projects are located in the `<pf_install>/sdk/plugin-src` directory.

Adapter and STS token-translator interfaces

The adapter and token-translator APIs enable PingFederate integration with IdPs or SPs. The APIs allow developers to build their own custom implementations for communicating authentication and security information between PingFederate and the enterprise environment.

Note: Token-translator interfaces are applicable only to PingFederate versions 6.0 and higher.

In addition to providing requisite runtime integration, an adapter or token translator also describes its configuration parameters to PingFederate; this enables the administrative console to render configuration screens with extensible validation.

Note: Suitable adapter or token-translator implementations for your deployment may already exist, or new implementations may be under development. Before developing your own custom solution, see the Ping Identity Downloads website for more information about currently 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 connection (for federation hub use cases), depending on the specified conditions. For example, an authentication selector may map internal corporate users to use one adapter, while it maps external non-corporate users to a different adapter.

Note: Authentication selector interfaces are applicable only to PingFederate versions 6.6 and higher.

Authentication electors are configurable UI plug-ins, allowing you to render custom configuration screens.

Custom data source interfaces

The custom data source API is a set of Java interfaces that enable PingFederate to integrate with data stores not covered by existing JDBC or LDAP drivers. This allows developers to retrieve attributes from a data source of their choice during attribute fulfillment for various use cases. Similar to the adapter API, custom data source plug-ins also provide much of the same UI configuration functionality.

Password credential validator interfaces

The password credential validator interfaces allow developers to define credential validators that are used to 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.

Note: Credential validator interfaces are applicable only to PingFederate versions 6.5 and higher.

Identity store provisioner interfaces

Identity Store Provisioners provide a mechanism for provisioning and deprovisioning users to external user stores. For example, a custom Identity Store Provisioner could be configured within an inbound provisioning IdP Connection to provision users using the SCIM protocol.

ⓘ **Note:** Identity Store Provisioner interfaces are applicable only to PingFederate versions 7.1 and higher.

Similar to the adapter API, Identity Store Provisioners are configurable UI plug-ins, allowing you to render custom configuration screens.

Ping Identity Global Client Services

If you need assistance in using the SDK, visit the Ping Identity Support website to see how we can help you with your application.

Get started with the SDK

The subsequent sections describe the directories and build components that comprise the SDK and provide instructions for setting up a development environment.

Directory structure

The PingFederate SDK directory (`<pf_install>/pingfederate/sdk`) contains the following:

- `plugin-src/` – The directory where you place your custom plug-in projects. This directory also contains example plug-in implementations showing a wide range of functionality. You may use these examples for developing 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` – This file contains properties used by the Ant build script, `build.xml`, to compile and deploy your custom components. Do not modify this file; use `build.local.properties` to override any properties, if needed.
- `build.local.properties` – Allows you to specify which project you want to build and define properties specific to your environment. The main use of this file is declaring 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.

Set up your project

About this task

To start developing your own plug-in:

Steps

1. Before you start, ensure you have the Java SDK and Apache Ant installed.
2. To create a new plug-in, create a new project directory in the `<pf_install_dir>/pingfederate/sdk/plugin-src` directory.
3. In the new project directory, create a subdirectory named `java`.

   This is where you place the Java source code for your implementation(s).

   Follow standard Java package and directory structure layout.

4. If your project depends on third-party libraries, create another subdirectory called `lib` and place the necessary JAR files in it.
5. The build script builds only one project at a time. Edit the `build.local.properties` file and set `target-plugin-name` to specify the name of the directory (under `<pf_install>/pingfederate/sdk/plugin-src`) that contains your project.
6. In `<pf_install>/pingfederate/sdk` run **`ant`** to display a list of available build targets:

```
[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
```

   Run the appropriate target to clean, build, or deploy your plug-in.

Note: Building the project with the `build.xml` included in the SDK is recommended because it packages the jars with additional metadata to make it discoverable by PingFederate. For detailed information, see *Build and deploy your project* on page 949.

Implementation guidelines

The following sections provide specific programming guidance for developing custom interfaces. Note that the information is not exhaustive—consult the Javadocs to find more details about interfaces discussed here and additional functionality.

Shared interfaces

All plug-in implementations generally invoke methods discussed in the subsequent sections.

Configurable plug-in

Any custom plug-in that requires UI settings is considered *configurable* and hence implements the `ConfigurablePlugin` interface. This ensures that PingFederate loads the plug-in instance with the correct configuration settings.

All plug-in types implement the `ConfigurablePlugin` interface and must define the following to enable configuration loading:

```
void configure(Configuration configuration)
```

During processing of a configurable plug-in instance, PingFederate calls the `ConfigurablePlugin.configure()` method and passes in a Configuration object. The Configuration object provides the plug-in 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. Once your implementation loads the configuration values, the plug-in instance can use them in other method calls.

Describable plug-in

Any plug-in that requires configuration screens in the PingFederate administrative console is considered a *describable* plug-in. Most plug-ins implement the `DescribablePlugin` interface to ensure that PingFederate renders the correct UI components based on the returned `PluginDescriptor`.

Adapter and custom data source plug-ins are a special case and do not implement the `DescribablePlugin` interface. However, they still return a plug-in descriptor

(`AuthnAdapterDescriptor` and `SourceDescriptor` respectively) and are still considered describable plug-ins.

All describable plug-ins must define a UI descriptor. Use one of the following methods to implement a UI descriptor, depending on the type of plug-in:

- For `DescribablePlugin`:

  ```
  PluginDescriptor getPluginDescriptor()
  ```

- For adapter plug-ins:

  ```
  AuthnAdapterDescriptor getAdapterDescriptor()
  ```

- For custom data source plug-ins:

  ```
  SourceDescriptor getSourceDescriptor()
  ```

In many cases, describable plug-ins return a subclass of `PluginDescriptor`, so the return type of the plug-in descriptor getters might be slightly different among plug-in implementations. Your plug-in implementation populates `PluginDescriptor` with `FieldDescriptors`, `FieldValidators`, and `Actions` and is presented as a set of UI components in the PingFederate administrative console.

Tip: Some plug-in types offer concrete descriptor implementations for developers. The Javadocs and examples provided with the SDK show which descriptor classes are available for each plug-in type. The examples also show you how to use `FieldDescriptors`, `FieldValidators`, and `Actions` directly to define your plug-in descriptor.

Implement an IdP adapter

You create an IdP adapter by implementing the `IdpAuthenticationAdapterV2` interface. The following Java packages are needed, at a minimum, for implementing this interface:

- `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 under *Shared interfaces* on page 934, you must define the following:

- Session Lookup
- Session Logout

IdP adapter session lookup

```
java.util.Map lookupAuthN(javax.servlet.http.HttpServletRequest req,
 javax.servlet.http.HttpServletResponse resp,
 java.lang.String partnerSpEntityId,
 AuthnPolicy authnPolicy,
 java.lang.String resumePath)
 throws AuthnAdapterException, java.io.IOException
```

PingFederate invokes the `lookupAuthN()` method of your IdP adapter to look up user-session information to handle a request. This method is invoked regardless of whether the request is for IdP- or SP-initiated SSO, an OAuth transaction, or direct IdP-to-SP adapter processing.

Note: The `IdpAuthenticationAdapterV2` interface provides an overloaded version of `lookupAuthN()` applicable to PingFederate versions 6.4 and higher. Use this interface if your adapter requires additional parameters from PingFederate. Refer to the `IdpAuthenticationAdapterV2` interface in the Javadocs for a complete list of available parameters.

In most implementations, a user's session information or a reference to it is communicated to PingFederate via the HttpServletRequest, which is passed to the `lookupAuthN()` method. For example, the user's session information can be passed in by the IdP application as a cookie or query parameter.

If the request from the user's browser does not contain the necessary information to identify the user, you can use the HttpServletResponse in various ways to retrieve the user's session data—for example, by creating a 302 redirect or presenting a web page asking for credentials. If your adapter implementation uses the HttpServletResponse to retrieve the user's session information, you must return the user's browser to the URL in the resumePath parameter set by the PingFederate runtime server and passed to this method. The resumePath is a relative URL signaling PingFederate that a user is continuing an SSO transaction that has already been initiated.

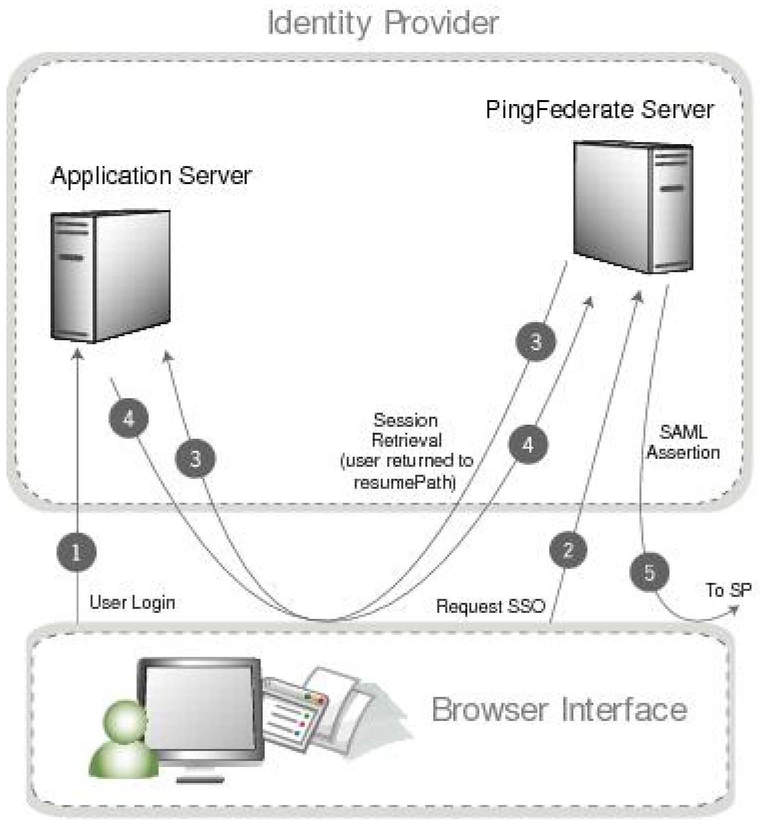
Tip: When creating a custom adapter, you can design it to render a template for processing and returning HTML to the user's browser using the `TemplateRendererUtil`. A sample `(template-render-adapter-example)` is included in the `sdk/plugin-src` directory of your PingFederate instance.

If your adapter implementation writes to the HttpServletResponse to retrieve the user's session data, we recommend that the browser return to the resumePath URL at all times, whether the retrieval succeeds or fails. Doing so ensures the adapter does not interrupt the “adapter chain” if it is used with the Composite Adapter. The Composite Adapter allows an administrator to “chain” together a selection of available adapter instances for a connection. At runtime, adapter chaining means that SSO requests are passed sequentially through each adapter instance until one or more authentication results are found for the user. If the browser is unable to return to the resumePath URL at all times, then it could interrupt the adapter chain causing unexpected results for the Composite Adapter.

For some authentication mechanisms, not all adapters can return the browser to the resumePath URL. Such adapters should not be used with the Composite Adapter's “Sufficient” chaining policy (see *Composite Adapter Configuration*).

Processing steps

The following diagram illustrates the request sequence of an IdP-initiated SSO scenario that uses the resumePath:



Processing steps

1. User logs in to a local application or domain through an identity-management system or some other authentication mechanism.
2. User clicks a link or otherwise requests access to a protected resource located in the 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. In this example, the adapter determines there is not enough information and redirects the browser to the application server to fetch additional session information.
4. The application server returns the session information and redirects the browser along with the returned information to resumePath URL.
5. PingFederate generates a SAML assertion and sends the browser with the SAML assertion to the SP's SAML gateway.

IdP adapter session logout

```
boolean  logoutAuthN(java.util.Map authnIdentifiers,
  javax.servlet.http.HttpServletRequest req,
  javax.servlet.http.HttpServletResponse resp,
  java.lang.String  resumePath)
  throws AuthnAdapterException, java.io.IOException
```

During 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.

Like the `lookupAuthN()` method, the `logoutAuthN()` method has access to the user's HttpServletRequest and HttpServletResponse objects. Use these objects to retrieve data about the user's session and to redirect the browser to an endpoint used to terminate the session at the application. Again, the resumePath parameter contains the URL to which the user is redirected to complete the SLO process.

Implement an SP adapter

You create an SP adapter by implementing the `SPAuthenticationAdapter` interface. The Java packages required are, at a minimum:

- `org.sourceid.saml20.adapter.sp.authn`
- `org.sourceid.saml20.adapter.gui`
- `org.sourceid.saml20.adapter.conf`

At a high level, in addition to the methods described under *Shared interfaces* on page 934, you must define the following:

- Session Creation
- Session Logout
- Account Linking (if configured in PingFederate for an IdP partner)

SP session creation

```
java.io.Serializable createAuthN(SsoContext ssoContext,
  javax.servlet.http.HttpServletRequest req,
  javax.servlet.http.HttpServletResponse resp,
  java.lang.String resumePath)
```

PingFederate invokes the `createAuthN()` method during the processing of an SSO request to establish a security context in the external application for the user. This method is similar to the `IdpAuthenticationAdapter.lookupAuthN()` method in terms of the objects passed to it and its support for asynchronous requests via the HttpServletResponse and resumePath parameters. This method also accepts an SsoContext object, which has access to information such as user attributes and the target destination URL.

SP adapter session logout

```
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
```

PingFederate invokes the `logoutAuthN()` method during an SLO request to terminate a user's session with the external application. The HttpServletResponse and resumePath objects are available to support scenarios where redirection of the user's browser is needed to an additional service to clean up any remaining sessions.

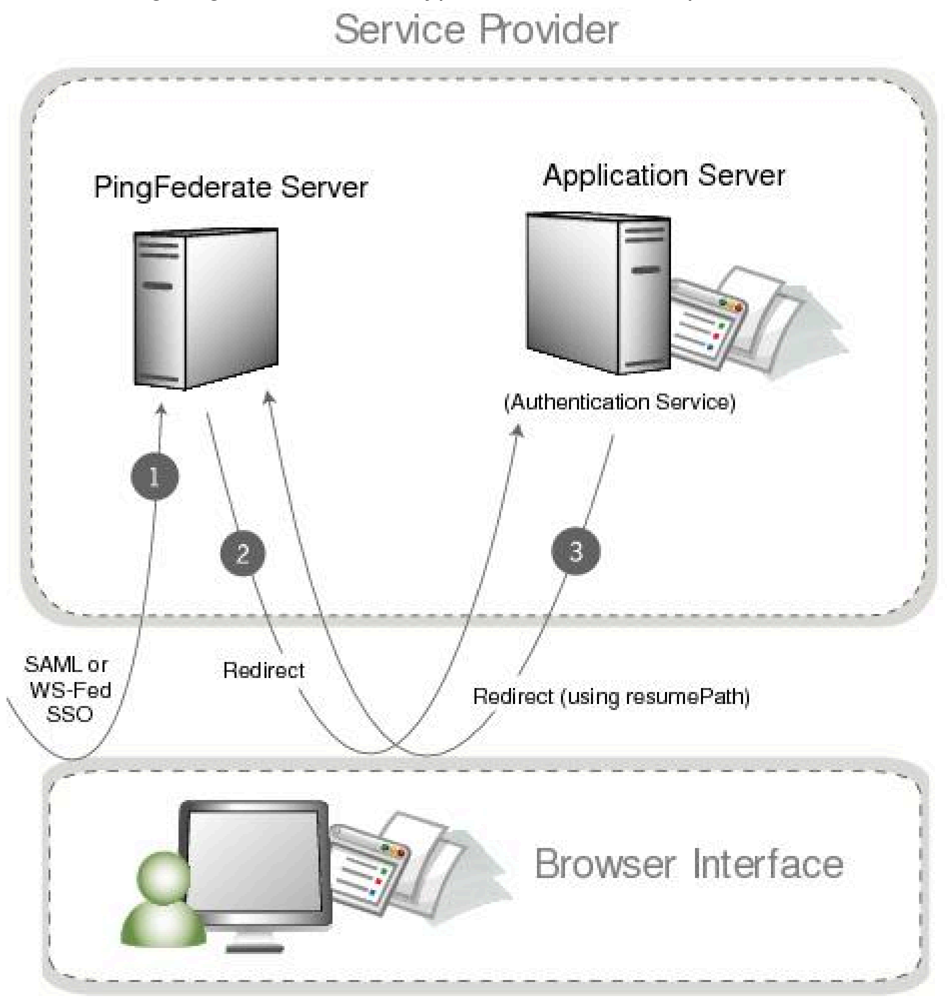
SP account linking

```
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
```

PingFederate invokes the `lookupLocalUserId()` method during an SSO request when the IdP connection is configured to use account linking but no account link for this user is yet established. Once the account link is set, PingFederate maintains this information until the user “defederates.” Defederation occurs when the user clicks a link redirecting him/her 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 String object returned from the `lookupLocalUserId()` method should be a local user identifier.

Implement a token processor

You create a token-processor implementation (for PingFederate 6.0 and higher) by implementing the `TokenProcessor` interface. The following Java packages are needed, at a minimum, for implementing this 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 interfaces* on page 934, you must define the method:

`TokenContext processToken(T token)`

PingFederate invokes the `processToken()` method during the processing of an STS request to perform necessary operations for determining the validity of a token. Type `T` must extend, at a minimum, the type `SecurityToken`. The type `BinarySecurityToken` is also available and may be used to represent custom security tokens that can be transported as Base64-encoded data.

Implement a token generator

You create a token-generator implementation (for PingFederate 6.0 and higher) by implementing the `TokenGenerator` interface. The following Java packages needed, at a minimum, for implementing this 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, described under *Shared interfaces* on page 934, you must define the method:

`SecurityToken generateToken(TokenContext attributeContext)`

PingFederate invokes the `generateToken()` method during the processing of an STS request to perform necessary operations for generation of a security token. The type `BinarySecurityToken` is available and may be used 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.

Implement an authentication selector

Authentication selectors allow PingFederate (version 6.6 and higher) to choose an appropriate authentication source, an 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, the following are the primary Java packages used:

- `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 interfaces* on page 934, you must define the following at a minimum:

- Context Selection
- Authentication Selector Callback

Context selection

```
AuthenticationSelectorContext selectContext(HttpServletRequest req,
 HttpServletResponse resp,
 Map<AuthenticationSourceKey, String> mappedAuthnSourcesNames,
 Map<String, Object> extraParameters,
 String resumePath)
```

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. The HttpServletRequest is available to evaluate cookies, parameters, headers, etc. to help determine which authentication source should be selected. The

HttpServletResponse is also available if the authentication selector requires user interaction to help determine the appropriate authentication source to select. 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 that should be used in conjunction with the resp object, such that the user's browser can be sent to this URL to resume the SSO workflow.

Once an authentication source is selected, an AuthenticationSelectorContext can be created to denote which authentication source to use. The selected authentication source can be referenced by its ID or by its context. The context is a name that decouples authentication selectors from the configured IDs.

Authentication selector callback

```
void callback(HttpServletRequest req,
HttpServletResponse resp,
Map authnIdentifiers,
AuthenticationSourceKey authenticationSourceKey,
AuthenticationSelectorContext authnSelectorContext);
```

PingFederate calls the `callback()` method after a selected authentication source is authenticated against. The callback() method allows authentication selectors to update resulting attributes, set cookies, or perform other custom functions.

Note: Writing content to the resp object in the callback() method is not supported, and doing so may result in unexpected behavior. Setting cookies is acceptable.

Implement a custom data source

Out of the box, PingFederate provides the capability of querying data sources for a variety of purposes using LDAP or JDBC interfaces. You can 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 perhaps 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 under *Shared interfaces* on page 934, you must define the following at a minimum:

- Connection Testing
- Available Fields Retrieval
- Data Source Query Handling

Data source connection testing

```
boolean testConnection()
```

When associating a custom data source with an IdP or 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`. Return `false` if your implementation cannot communicate with the data store. 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 could 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. The administrator can then select the attributes from the data source and map 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 that should be populated with data. The method may also receive a filterConfiguration object populated with a list of fields. Each field contains a name/value pair that is 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.

You 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.

Once all the 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 that was passed into the method and the values are the attributes.

Implement a password credential validator

Password credential validators allow PingFederate administrators to define a centralized location for username/password validation, allowing validator instances to be referenced by various PingFederate configurations.

To implement a custom password credential validator, the following Java packages need to be imported:

- `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 interfaces* on page 934, you must define the following at a minimum:

```
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, then an AttributeMap is returned containing at least one entry representing the principal. If the credentials are invalid, then `null` or an empty map is returned. A PasswordValidationException is thrown if the plug-in was unable to validate the credentials (for example, due to an offline host or network problems).

To enable change password in a password credential validator, implement the `com.pingidentity.sdk.password.ChangeablePasswordCredential` interface.

To enable password reset in a password credential validator, implement the `com.pingidentity.sdk.password.ResettablePasswordCredential` interface.

ⓘ **Note:** Depending on your password management system, additional system configuration may be necessary to enable password changes—for example, passwords can be changed in Active Directory only if SSL is enabled.

Implement an identity store provisioner

You create an Identity Store Provisioner by implementing the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interface.

Both interfaces support provisioning and deprovisioning users, and optionally groups, to an external user store. The `IdentityStoreProvisionerWithFiltering` supports list/query and filtering, whereas the `IdentityStoreProvisioner` does not. For more information about list/query and filtering, see *3.2.2 List/Query Resources* and *3.2.2.1 Filtering* in SCIM Specification (www.simplecloud.info/specs/draft-scim-api-01.html).

ⓘ **Note:** As of PingFederate 7.3, the `IdentityStore`*`User`*`Provisioner` interface has been deprecated. Developers are encouraged to implement either the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interface.

Implement the IdentityStoreProvisionerWithFiltering interface

Implement the `IdentityStoreProvisionerWithFiltering` interface to provision and deprovision users, and optionally 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.

The following Java packages are needed, at a minimum, for implementing this interface:

- `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* on page 945).

For each Identity Store Provisioner implementation, in addition to the methods described under *Shared interfaces* on page 934, 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, for example inbound provisioning. This method is responsible for creating the user in the user store managed by the Identity Store Provisioner.

The `CreateUserRequestContext` will contain all information needed to fulfill the request (for example, user attributes). If the user was successfully provisioned, a `UserResponseContext` should be returned and contain the user attributes used to provision the user. An `IdentityStoreException` should be thrown if an error occurred during the creation process. See `com.pingidentity.sdk.provision.exception` package for different 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, for example Inbound Provisioning. This method is responsible for retrieving user data from the user store managed by the Identity Store Provisioner.

The `ReadUserRequestContext` will contain all information needed to fulfill the request (for example, a user ID). If the user data was successfully retrieved, a `UserResponseContext` should be returned and contain the user attributes for the user. 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.

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, for example inbound provisioning. This method is responsible for retrieving user data from the user store managed by the Identity Store Provisioner.

Note: The `readUsers` method is applicable only to the `IdentityStoreProvisionerWithFiltering` interface; it does not apply to the `IdentityStoreProvisioner` interface.

The `ReadUsersRequestContext` will contain all information needed to fulfill the request (for example, a filter). If the user data was successfully retrieved, a `UsersResponseContext` should be returned and contain the user attributes satisfying the filter. 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 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, for example inbound provisioning. This method is responsible for updating the user in the user store managed by the Identity Store Provisioner.

The `UpdateUserRequestContext` will contain all information needed to fulfill the request (for example, user attributes). If the user data was successfully updated, a `UserResponseContext` should be returned containing the user's updated attributes. An `IdentityStoreException` should be thrown if an error occurred during the update process. See `com.pingidentity.sdk.provision.exception` package for different 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 is responsible for deprovisioning the user in the user store managed by the Identity Store Provisioner.

The `DeleteUserRequestContext` will contain all information needed to fulfill the request (for example, a user ID). An `IdentityStoreException` should be thrown if an error occurred during the deprovision process. See `com.pingidentity.sdk.provision.exception` package for different exceptions that can be thrown.

Note: The plugin implementation for delete MAY 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 this `isGroupProvisioningSupported()` method to return true if group provisioning is supported by your Identity Store Provisioner or false otherwise. An `IdentityStoreException` should be thrown if an error occurred during the query process. See `com.pingidentity.sdk.provision.exception` package for different 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, for example inbound provisioning. This method is responsible for creating the group in the user store managed by the Identity Store Provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `CreateGroupRequestContext` will contain all information needed to fulfill the request (for example, group attributes). If the group was successfully provisioned, a `GroupResponseContext` should be returned and contain the group attributes used to provision the group. An `IdentityStoreException` should be thrown if an error occurred during the creation process. See `com.pingidentity.sdk.provision.exception` package for different 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, for example inbound provisioning. This method is responsible for retrieving group data from the user store managed by the Identity Store Provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `ReadGroupRequestContext` will contain all information needed to fulfill the request (for example, group ID). If the group data was successfully retrieved, a `GroupResponseContext` should be returned and contain the group attributes for the group. 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.

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, for example inbound provisioning. This method is responsible for retrieving 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 is applicable 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, for example inbound provisioning. This method is responsible for updating the group in the user store managed by the Identity Store

Provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `UpdateGroupRequestContext` will contain all information needed to fulfill the request (for example, group attributes). If the group data was successfully updated, a `GroupResponseContext` should be returned containing the group's updated attributes. An `IdentityStoreException` should be thrown if an error occurred during the update process. See `com.pingidentity.sdk.provision.exception` package for different 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 is responsible for deprovisioning the group in the user store managed by the Identity Store Provisioner if the `isGroupProvisioningSupported()` returns true; otherwise, it should throw `NotImplementedException`.

The `DeleteGroupRequestContext` will contain all information needed to fulfill the request (for example, a group ID). An `IdentityStoreException` should be thrown if an error occurred during the deprovision process. See `com.pingidentity.sdk.provision.exception` package for different exceptions that can be thrown.

Implement the IdentityStoreUserProvisioner interface

Note: The `IdentityStoreUserProvisioner` interface has been deprecated since PingFederate 7.3. Developers are encouraged to implement either the `IdentityStoreProvisionerWithFiltering` or `IdentityStoreProvisioner` interface.

Implement the `IdentityStoreUserProvisioner` interface to provision and deprovision users to an external user store.

Tip: The `IdentityStoreUserProvisioner` interface does not provision or deprovision groups. For group support, see *Implement the IdentityStoreProvisionerWithFiltering interface* on page 943.

The following Java packages are needed, at a minimum, for implementing this 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 interfaces* on page 934, you must implement the following:

- 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, for example inbound provisioning. This method is responsible for creating the user in the user store managed by the Identity Store Provisioner.

The `CreateUserRequestContext` will contain all information needed to fulfill the request (for example, user attributes). If the user was successfully provisioned, a `UserResponseContext` should be returned and contain the user attributes used to provision the user. An `IdentityStoreException` should be thrown if an error occurred during the creation process. See `com.pingidentity.sdk.provision.exception` package for different 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, for example Inbound Provisioning. This method is responsible for retrieving user data from the user store managed by the Identity Store Provisioner.

The `ReadUserRequestContext` will contain all information needed to fulfill the request (for example, a user ID). If the user data was successfully retrieved, a `UserResponseContext` should be returned and contain the user attributes for the user. 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 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, for example inbound provisioning. This method is responsible for updating the user in the user store managed by the Identity Store Provisioner.

The `UpdateUserRequestContext` will contain all information needed to fulfill the request (for example, user attributes). If the user data was successfully updated, a `UserResponseContext` should be returned containing the user's updated attributes. An `IdentityStoreException` should be thrown if an error occurred during the update process. See `com.pingidentity.sdk.provision.exception` package for different 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 is responsible for deprovisioning the user in the user store managed by the Identity Store Provisioner.

The `DeleteUserRequestContext` will contain all information needed to fulfill the request (for example, a user ID). An `IdentityStoreException` should be thrown if an error occurred during the deprovision process. See `com.pingidentity.sdk.provision.exception` package for different exceptions that can be thrown.

Note: The plugin implementation for delete MAY 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`.

Build and deploy your project

To build and deploy your project, you can choose to use the provided Apache Ant script or another build utility.

Build and deploy with Ant

About this task

The PingFederate Java SDK comes with an Apache Ant build script that makes building and deploying your project simple.

Steps

1. Edit the `build.local.properties` file and set the target-plugin.name property to the name of your project subdirectory (see *Directory structure* on page 933).

   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 jars, 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.

   To clean the project, enter:

   ```
   ant clean-plugin
   ```

   To compile the project, enter:

   ```
   ant compile-plugin
   ```

   To compile the project and create a JAR, enter:

   ```
   ant jar-plugin
   ```

   The SDK creates deployment descriptor(s) in the `PF_INF` directory and places it in a JAR. The descriptor tells PingFederate what plug-in implementations are contained in the JAR.

   The compiled class files and the deployment descriptor(s) 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, create a JAR, and deploy the project to PingFederate, enter:

   ```
   ant deploy-plugin
   ```

   This build target performs the steps described above and deploying any JAR files found in the `lib` directory of your subproject.

   Note: To deploy your plug-in 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.

Build and deploy manually

About this task

To build your project with another build utility, you must take some prerequisite steps to create the deployment descriptors for each of your plug-ins. The deployment descriptor files allow PingFederate to discover your plug-ins.

Create deployment descriptors

Steps

1. In your project, create a new directory called `PF-INF`. This directory must be at the root of your JAR file, similar to `META-INF`.
2. Inside `PF-INF` create the appropriate text file(s) for each type of plug-ins you created:

Plug-in Type	Filename
IdP Adapter	idp-authn-adapters
SP Adapter	sp-authn-adapters
Custom Data Source	custom-drivers
Token Processor	token-processors

Plug-in Type	**Filename**
Token Generator	token-generators
Authentication Selector	authentication-selectors
Password Credential Validator	password-credential-validators
Identity Store Provisioner	identity-store-provisioners

3. In each text file created, specify the fully qualified class name of each plug-in that implements the corresponding plug-in interface. Place each class name on a separate line.

Build your project manually

Steps

- To compile your project, you need to 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`
- To create a JAR, simply archive the compiled class files along with the deployment descriptor(s) using your build tool. The deployment descriptors must be in the `PF-INF` directory, located at the root of the JAR.

Deploy your project

Steps

- To deploy your plug-in, simply copy the JAR file and any third-party JAR files into the `<pf_install>/pingfederate/server/default/deploy` directory of the PingFederate installation.

Logging

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 SP adapter contained in the directory `sdk/adapters-src/sp-adapter-example` shows how to use a logger for your adapter.

PingFederate Upgrade Guide

Depending on the existing Ping Identity® installation, you (the administrator) can upgrade using an installer or the Upgrade Utility, which automatically migrates existing PingFederate 6.0 (or a more recent) configuration to version 9.3.

Both the platform-specific installers and the Upgrade Utility create a new installation based on the new product distribution ZIP file, 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.

Note: If you are upgrading from PingFederate 5.3 (or an earlier version), contact Support for more information.

Intended audience

The upgrade tools and this guide are intended for system administrators with significant knowledge of the current PingFederate configuration and enterprise deployment.

Upgrade overview

Fully upgrading your existing PingFederate installation involves initial preparation ahead of the upgrade process. Then afterward, some manual copying may be needed for customized files or specific deployment scenarios.

ⓘ **Important:** We recommend that you upgrade your test environment and verify that the new installation meets your expectations before upgrading your production environment. After you complete the upgrade process successfully, you may create a backup of your previous installation and remove it from your server.

Additionally, end users may experience service disruptions as you upgrade your PingFederate environment. As needed, schedule a maintenance window to perform the upgrade.

Upgrade tools

Depending on the source installation, you may upgrade PingFederate by using a platform-specific installer or the Upgrade Utility. The conditions are described as follows:

Operating system	Source version	Source installation medium	Possible upgrade paths
Microsoft Windows	6.x through 9.x	PingFederate product distribution ZIP file	PingFederate Upgrade Utility
	8.x through 9.x	PingFederate installer for Windows	PingFederate installer for Windows, or PingFederate Upgrade Utility
Red Hat Enterprise Linux	6.x through 9.x	PingFederate product distribution ZIP file	PingFederate install script, or PingFederate Upgrade Utility
	8.x through 9.x	PingFederate install script	PingFederate install script, or PingFederate Upgrade Utility
Other qualified operating systems	6.x through 9.x	PingFederate product distribution ZIP file	PingFederate Upgrade Utility

You may download the PingFederate platform-specific installers from the PingFederate Downloads website.

Both the platform-specific installers and the Upgrade Utility create a new installation based on the new product distribution ZIP file, 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. While the tools do not upgrade the integration kits automatically, you may download newer versions by visiting the Ping Identity Downloads website and upgrade the integration kits manually.

ⓘ **Note:** Documentation for integration kits are available from the Ping Identity *PingFederate documentation* website.

Preparation

Upgrading your existing PingFederate installation involves the following preparation tasks:

- 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.
- If you have not already done so, obtain a new license key as needed.
- If you have not updated the Java runtime to version 8 or 11 on your PingFederate servers, you must do so.

  When you upgrade the Java runtime, be sure to modify the previously defined paths for the system JAVA_HOME and PATH environment variables.

  ⓘ **Important:** PingFederate versions prior to 7.2 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 PingFederate on Windows

About this task

On a Windows server, depending on the source installation, you may upgrade your PingFederate installation from version 6.0 (or a more recent version) to version 9.3 by using the PingFederate installer for Windows or the PingFederate Upgrade Utility at the command prompt. The conditions are described as follows:

Source Version	Source installation medium	Possible upgrade paths on a Windows server
6.x through 9.x	PingFederate product distribution ZIP file	PingFederate Upgrade Utility
8.x through 9.x	PingFederate installer for Windows	PingFederate installer for Windows, or PingFederate Upgrade Utility

ⓘ **Note:** If you are upgrading from PingFederate 5.3 (or an earlier version), contact Support for more information.

Both the platform-specific installers and the Upgrade Utility create a new installation based on the new product distribution ZIP file, 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.

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. While the tools do not upgrade the integration kits automatically, you may download newer versions by visiting the Ping Identity Downloads website and upgrade the integration kits manually.

Important: We recommend that you upgrade your test environment and verify that the new installation meets your expectations before upgrading your production environment. After you complete the upgrade process successfully, you may create a backup of your previous installation and remove it from your server.

Additionally, end users may experience service disruptions as you upgrade your PingFederate environment. As needed, schedule a maintenance window to perform the upgrade.

Note: The PingFederate installer for Windows does not support custom mode. If you want to override the newer default security settings or upgrade the OpenToken Adapter, use the PingFederate Upgrade Utility.

Steps

- To use the platform-specific installer, download it from https://www.pingidentity.com/en/products/downloads/pingfederate/platform.html.
- To use the Upgrade Utility, download the latest version of PingFederate and its Upgrade Utility at https://www.pingidentity.com/en/products/downloads/pingfederate/upgrade.html.

Upgrade PingFederate using the installer

About this task

Follow these steps to upgrade your PingFederate server using the PingFederate platform-specific installer.

Important: If you are upgrading a clustered PingFederate environment, start with the console node, and then follow the additional steps to upgrade the engine nodes.

Steps

1. Ensure that you are logged on to your server with appropriate privileges to install and run an application.
2. Run the PingFederate installer for Windows.

3. Follow the wizard to upgrade PingFederate.

   Errors are rare but may include such input/output problems as missing or malformed configuration files in the source installation. If the upgrade tool reports an error, review the error messages in the `upgrade-full.log` file.

   > **Note:** You may 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.

   > **Note:** If you are upgrading a clustered PingFederate environment, stop PingFederate to safeguard your clustered environment from running different versions of PingFederate during the upgrade process.

4. If you are upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

   > **Note:** End users may experience service disruptions as you upgrade your PingFederate environment.

5. Start the new PingFederate installation and open the administrative console.

   If you are upgrading a clustered PingFederate environment, start the new PingFederate on the console node.
6. Follow the on-screen instructions to upload your license file for the new version.
7. If you are upgrading a clustered PingFederate environment, start the new installation on each engine node, and then ensure all nodes are shown on the **System**# **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **System**# **Cluster Management** screen.

Upgrade PingFederate using the Upgrade Utility on Windows

About this task

Follow these steps to upgrade your PingFederate server using the PingFederate Upgrade Utility at the command prompt.

> **Important:** If you are upgrading a clustered PingFederate environment, start with the console node, and then follow the additional steps to upgrade the engine nodes.

Steps

1. Unzip the Upgrade Utility ZIP file into a directory on the same machine running PingFederate.
2. Stop PingFederate.
3. At the command prompt, change the current directory to the `bin` directory of the Upgrade Utility and execute the following command:

   ```
   upgrade <pf_install_source> <outputDir> <pingfederateZip> [-c]
   ```

   where:

   `<pf_install_source>`

The full or relative path of the base directory where the existing PingFederate software (`pingfederate`) is installed.

> **Note:** The `pingfederate` subdirectory *must* exist by that name for the Upgrade Utility to function correctly.

`<outputDir>`

The full or relative path of the directory that will contain the new PingFederate base directory.

`<pingfederateZip>`

The full or relative path and file name for the current distribution.

`-c`

The optional parameter to run the tool in custom mode, which allows administrators to override newer default security settings (if any) and to upgrade the program files for the OpenToken Adapter.

After a moment, the command prompt displays messages indicating upgrade progress. The process is complete when this message appears:

```
Upgrade completed with [N] errors and [N] warnings
```

If there are errors, scroll up the command window to see them and correct indicated problems. Errors during the upgrade should be rare but may include such input/output problems as missing or malformed configuration files in the source installation. The messages, including any errors, are also logged to the `upgrade.log` file in the Upgrade Utility base directory.

> **Note:** You may rerun the tool as many times as needed to correct any problems.

4. If you are upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

> **Note:** End users may experience service disruptions as you upgrade your PingFederate environment.

5. Start the new PingFederate installation and open the administrative console.

   If you are upgrading a clustered PingFederate environment, start the new PingFederate on the console node.
6. Follow the on-screen instructions to upload your license file for the new version.
7. If you are upgrading a clustered PingFederate environment, start the new installation on each engine node, and then ensure all nodes are shown on the **System# Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **System# Cluster Management** screen.
9. If PingFederate is running as a service, re-install the service.
   a. Remove the existing PingFederate service (see *Uninstall PingFederate from a Windows server* on page 76).
   b. Install the new PingFederate service (see *Install PingFederate service on Windows manually* on page 72).

   For a clustered PingFederate environment, re-install the PingFederate service on all nodes.

Upgrade PingFederate on Red Hat Enterprise Linux

About this task

On a Red Hat Enterprise Linux server, you may upgrade your PingFederate installation from version 6.0 (or a more recent version) to version 9.3 by using the PingFederate install script or the PingFederate Upgrade Utility. (They are both command-line tools.)

Source Version	Source installation medium	Possible upgrade paths on a Linux server
6.x through 9.x	PingFederate product distribution ZIP file	PingFederate install script, or PingFederate Upgrade Utility
8.x through 9.x	PingFederate install script	PingFederate install script, or PingFederate Upgrade Utility

Note: If you are upgrading from PingFederate 5.3 (or an earlier version), contact Support for more information.

Both the platform-specific installers and the Upgrade Utility create a new installation based on the new product distribution ZIP file, 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.

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. While the tools do not upgrade the integration kits automatically, you may download newer versions by visiting the Ping Identity Downloads website and upgrade the integration kits manually.

Important: We recommend that you upgrade your test environment and verify that the new installation meets your expectations before upgrading your production environment. After you complete the upgrade process successfully, you may create a backup of your previous installation and remove it from your server.

Additionally, end users may experience service disruptions as you upgrade your PingFederate environment. As needed, schedule a maintenance window to perform the upgrade.

Steps

- To use the platform-specific installer, download it from https://www.pingidentity.com/en/products/downloads/pingfederate/platform.html.
- To use the Upgrade Utility, download the latest version of PingFederate and its Upgrade Utility at https://www.pingidentity.com/en/products/downloads/pingfederate/upgrade.html.

Upgrade PingFederate using the install script

About this task

Follow these steps to upgrade your PingFederate server using the PingFederate platform-specific installer.

Important: If you are upgrading a clustered PingFederate environment, start with the console node, and then follow the additional steps to upgrade the engine nodes.

Steps

1. Ensure that you are logged on to your server with appropriate privileges to install and run an application.
2. Run the PingFederate install script on the command line and follow the prompt to upgrade PingFederate.

   (The command is available from the download website.)

   Errors are rare but may include such input/output problems as missing or malformed configuration files in the source installation. If the upgrade tool reports an error, review the error messages in the `upgrade-full.log` file.

   > **Note:** You may 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.

   > **Note:** If you are upgrading a clustered PingFederate environment, stop PingFederate to safeguard your clustered environment from running different versions of PingFederate during the upgrade process.
3. If you are upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

   > **Note:** End users may experience service disruptions as you upgrade your PingFederate environment.
4. Start the new PingFederate installation and open the administrative console.

   If you are upgrading a clustered PingFederate environment, start the new PingFederate on the console node.
5. Follow the on-screen instructions to upload your license file for the new version.
6. If you are upgrading a clustered PingFederate environment, start the new installation on each engine node, and then ensure all nodes are shown on the **System**# **Cluster Management** screen.
7. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **System**# **Cluster Management** screen.

Upgrade PingFederate using the Upgrade Utility on Red Hat Enterprise Linux

About this task

Follow these steps to upgrade your PingFederate server using the PingFederate Upgrade Utility on the command line.

> **Important:** If you are upgrading a clustered PingFederate environment, start with the console node, and then follow the additional steps to upgrade the engine nodes.

Steps

1. Unzip the Upgrade Utility ZIP file into a directory on the same machine running PingFederate.
2. Stop PingFederate.
3. On the command line, change the current directory to the `bin` directory of the Upgrade Utility and execute the following command:

```
                   ./upgrade.sh <pf_install_source> <outputDir>
  <pingfederateZip> [-c]
```

where:

`<pf_install_source>`

The full or relative path of the base directory where the existing PingFederate software (`pingfederate`) is installed.

> **Note:** The `pingfederate` subdirectory *must* exist by that name for the Upgrade Utility to function correctly.

`<outputDir>`

The full or relative path of the directory that will contain the new PingFederate base directory.

`<pingfederateZip>`

The full or relative path and file name for the current distribution.

`-c`

The optional parameter to run the tool in custom mode, which allows administrators to override newer default security settings (if any) and to upgrade the program files for the OpenToken Adapter.

After a moment, the command prompt displays messages indicating upgrade progress. The process is complete when this message appears:

```
Upgrade completed with [N] errors and [N] warnings
```

If there are errors, scroll up the command window to see them and correct indicated problems. Errors during the upgrade should be rare but may include such input/output problems as missing or malformed configuration files in the source installation. The messages, including any errors, are also logged to the `upgrade.log` file in the Upgrade Utility base directory.

> **Note:** You may rerun the tool as many times as needed to correct any problems.

4. If you are upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

> **Note:** End users may experience service disruptions as you upgrade your PingFederate environment.

5. Start the new PingFederate installation and open the administrative console.

   If you are upgrading a clustered PingFederate environment, start the new PingFederate on the console node.
6. Follow the on-screen instructions to upload your license file for the new version.
7. If you are upgrading a clustered PingFederate environment, start the new installation on each engine node, and then ensure all nodes are shown on the **System**# **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **System**# **Cluster Management** screen.

9. If PingFederate is running as a service, re-configure the service.

PingFederate systemd service

Edit the PingFederate systemd unit file and reconfigure the PingFederate service (see *step 7* in *Install PingFederate service on Linux manually* on page 73).

PingFederate SysV initialization script

Edit the PingFederate SysV initialization script and reconfigure the PingFederate service (see *step 8* in *Install PingFederate service on Linux manually* on page 73).

Upgrade PingFederate on other operating systems

About this task

On a Linux server, you may upgrade your PingFederate installation from version 6.0 (or a more recent version) to version 9.3 by using the PingFederate Upgrade Utility. (They are both command-line tools.)

Refer to *System requirements* on page 60 for a list of qualified Linux operating systems.

Note: If you are upgrading from PingFederate 5.3 (or an earlier version), contact Support for more information.

The Upgrade Utility creates a new installation based on the new product distribution ZIP file, and then copies the relevant files and property values from the existing installation (the source) to the new installation (the target). As a result, the Upgrade Utility does not affect the source installation.

The Upgrade Utility also copies the program files for the deployed Adapters, Connectors, and Token Translators (the *integration kits* in general) from the source installation to the target installation. While the utility does not upgrade the integration kits automatically, you may download newer versions by visiting the Ping Identity Downloads website and upgrade the integration kits manually.

Important: We recommend that you upgrade your test environment and verify that the new installation meets your expectations before upgrading your production environment. After you complete the upgrade process successfully, you may create a backup of your previous installation and remove it from your server.

Additionally, end users may experience service disruptions as you upgrade your PingFederate environment. As needed, schedule a maintenance window to perform the upgrade.

Steps

- To use the Upgrade Utility, download the latest version of PingFederate and its Upgrade Utility at https://www.pingidentity.com/en/products/downloads/pingfederate/upgrade.html.

Upgrade PingFederate using the Upgrade Utility on other operating systems

About this task

Follow these steps to upgrade your PingFederate server using the PingFederate Upgrade Utility on the command line.

Refer to *System requirements* on page 60 for a list of qualified Linux operating systems.

ⓘ **Important:** If you are upgrading a clustered PingFederate environment, start with the console node, and then follow the additional steps to upgrade the engine nodes.

Steps

1. Unzip the Upgrade Utility ZIP file into a directory on the same machine running PingFederate.
2. Stop PingFederate.
3. On the command line, change the current directory to the `bin` directory of the Upgrade Utility and execute the following command:

   ```
                       ./upgrade.sh <pf_install_source> <outputDir>
   <pingfederateZip> [-c]
   ```

   where:

   `<pf_install_source>`

   The full or relative path of the base directory where the existing PingFederate software (`pingfederate`) is installed.

   ⓘ **Note:** The `pingfederate` subdirectory *must* exist by that name for the Upgrade Utility to function correctly.

   `<outputDir>`

   The full or relative path of the directory that will contain the new PingFederate base directory.

   `<pingfederateZip>`

   The full or relative path and file name for the current distribution.

   `-c`

   The optional parameter to run the tool in custom mode, which allows administrators to override newer default security settings (if any) and to upgrade the program files for the OpenToken Adapter.

   After a moment, the command prompt displays messages indicating upgrade progress. The process is complete when this message appears:

   ```
   Upgrade completed with [N] errors and [N] warnings
   ```

   If there are errors, scroll up the command window to see them and correct indicated problems. Errors during the upgrade should be rare but may include such input/output problems as missing or malformed configuration files in the source installation. The messages, including any errors, are also logged to the `upgrade.log` file in the Upgrade Utility base directory.

   ⓘ **Note:** You may rerun the tool as many times as needed to correct any problems.

4. If you are upgrading a clustered PingFederate environment, repeat from step 1 to upgrade PingFederate on each engine node.

ⓘ **Note:** End users may experience service disruptions as you upgrade your PingFederate environment.

5. Start the new PingFederate installation and open the administrative console.

   If you are upgrading a clustered PingFederate environment, start the new PingFederate on the console node.
6. Follow the on-screen instructions to upload your license file for the new version.
7. If you are upgrading a clustered PingFederate environment, start the new installation on each engine node, and then ensure all nodes are shown on the **System# Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **System# Cluster Management** screen.
9. If PingFederate is running as a service, re-configure the service.

   PingFederate systemd service

   Edit the PingFederate systemd unit file and reconfigure the PingFederate service (see *step 7* in *Install PingFederate service on Linux manually* on page 73).

   PingFederate SysV initialization script

   Edit the PingFederate SysV initialization script and reconfigure the PingFederate service (see *step 8* in *Install PingFederate service on Linux manually* on page 73).

Custom mode

The custom-mode feature in the Upgrade Utility (invoked with the `-c` option on the command line) allows you to override several newer default security settings (new, depending on which PingFederate version is currently running). 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, if applicable.

Security defaults

In general, using the new security defaults is highly recommended and should not cause significant issues for most PingFederate installations. The newer 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. If you want to see which cipher suites are commented out, choose yes (`y`) when prompted on whether to use the new defaults. Then, after the upgrade is complete, refer to either 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`
- Disabling SQLFilter as of PingFederate version 6.2.
- Disabling the sending of PingFederate session cookies by way of the unsecure hypertext transfer protocol. As of version 6.5, the session cookie is configured to be sent only using HTTPS.

Adapter upgrade

Upgrading the OpenToken Adapter from an earlier version is also recommended and will not normally require any follow-on configuration changes.

- If your existing installation uses a version of the OpenToken Adapter prior to 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 prior to 2.5.1, we recommend that you redeploy agent configuration files, if applicable, as well as any new agent libraries contained in recent versions of PingFederate integration kits and other plug-ins that use `OpenToken`.

Starting in PingFederate 7.2, the LDAP Java Adapter is no longer supported. This adapter was deprecated in PingFederate 6.6 and replaced by the LDAP Username Password Credential Validator (PCV), which can be used with the HTML Form Adapter or HTTP Basic Adapter.

Review post-upgrade tasks

About this task

Review the subsequent topics for post-upgrade tasks.

It is also important to perform runtime tests to ensure the new PingFederate installation fulfills your existing use cases.

Copy customized files or settings

User-facing screens

If you have modified any Velocity templates for user-facing screens and you wish to preserve the customized user experience, you must migrate your custom changes to the new installation manually for each server node. The templates are located in the `<pf_install>/pingfederate/server/default/conf/template` directory.

CAUTION: Supporting CSS and image file names were 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.

Example: `<link rel="..." href="assets/css/screen.1.css"/>`

Likewise, add `.1` to any references in the copied templates to the installed image files contained in the `assets/images` directory.

Example: `<img src="assets/images/green_check.1.png"/>`

Email notifications

Similarly, if you have modified the email notification templates prior to version 8.2 and you wish to preserve the customized user experience, you must migrate your custom changes to the new HTML-based templates manually for each server node. The plain text templates (`message-template-*.txt`) can be found in the `<pf_install>/pingfederate/server/default/conf` directory in the source installation. The new HTML-based templates are located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory with the same file naming convention but an `.html` file extension.

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 version 6.9 (or a more recent), you must merge any changes you have made to the relevant files in the Jetty configuration directory, `<pf_install>/pingfederate/etc`. Default values or the structure of these files may change with new releases, so it is important to selectively merge your changes.

If you are upgrading from version 6.8 through 6.0, first identify any changes made to the JBoss configuration, then make corresponding changes for the newer Jetty configuration.

For example, a commonly modified file prior to version 6.9 might be the `jboss-service.xml` file located in the `<pf_install>/pingfederate/server/default/deploy/jetty.sar/META-INF` directory. When changes are identified, 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`.

size-limits.conf

Prior to version 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 following data sets:

- Inter-request state information—the state information between the redirects to complete a request
- Adapter session-state data—the state information (along with the associated attributes and their values, if any) maintained (or used) by the adapters.

PingFederate 8.4.2 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 help 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 have 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 have 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 *Configure AS settings* on page 379.

Configuration files in the `config-store` directory

If you have 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 such setting values to the new installation.

Note: The upgrade tools do *not* copy comments from the existing installation to the new installation.

On rare occasions that you have removed a setting or a block of settings from a configuration file in the `config-store` directory, the upgrade tools preserves your changes by removing such setting or block of settings from the new installation and records the removals in its log file. If you wish to re-add such 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 file and make your changes.

Other configuration files

Other configuration files stored in the `<pf_install>/pingfederate/server/default/conf` directory, such as the `cluster-*.conf` files for advanced clustering deployment architecture or the `tcp.xml` for dynamic discovery, are *not* migrated automatically. To preserve your custom settings (if any), create a backup of the current configuration files and merge your changes to the current files.

Furthermore, if you have previously customized JVM options in the `run.bat`, `run.sh`, or `PingFederateService.conf` files, instead of updating these files, manually migrate your JVM options to the `jvm-memory.options` file located in the `<pf_install>/pingfederate/bin` directory. For more information, see *Fine-tune JVM options* on page 985 and *memoryoptions and upgrade* on page 981

Review database changes

Occasionally, PingFederate introduces database-related changes, such as adding a new table, modifying an existing table, or updating connection pool library, for the purpose of product improvement.

Additionally, neither the Upgrade Utility nor the platform-specific installers 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 is re-initialized from the provisioning source.

If your PingFederate environment connects to one or more database servers, review the following information and make changes accordingly.

Provisioning data store 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 has since been resolved in version 9.0.2.

If you are upgrading from version 8.4.4 (or an earlier version) or 9.0.2 to version 9.3, 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 version 9.3, use the following **`provmgr`** commands to reset the provisioning data store on the upgraded installation:

1. Get a list of provisioning channel IDs:

   ```
   provmgr --show-channels
   ```
2. Reset the provisioning data store 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 does not matter.)

The **`provmgr`** command-line tool is located in the `<pf_install>/pingfederate/bin` directory: `provmgr.bat` for Windows and `provmgr.sh` for Linux. For more information about the **`provmgr`** command-line tool, see *Outbound provisioning CLI* on page 255.

Security enhancement in JDBC data store queries

A security enhancement has been made in PingFederate 9.0 to safeguard JDBC data store queries against back-end SQL injection attacks. This protection is enabled for all new installations. For upgrades, you can enable this protection by modifying the `org.sourceid.common.SqlFilterManager.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

To enable this security enhancement:

1. Edit the `org.sourceid.common.SqlFilterManager.xml` file.

2. Set the <item name="enableSqlFilters"/> element value to `true`; for 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.

   If you have a clustered PingFederate environment:

   a. Perform the steps above on the console node.
   b. Sign on to the PingFederate administrative console.
   c. Go to the **System**# **Cluster Management** screen.
   d. Click **Replicate Configuration** to push this change to all engine nodes.
5. Verify your use cases to make sure your search filters return the expected results.

An improved index in the database table for OAuth clients

Applicable only to customers who had configured PingFederate to store OAuth clients on a database server.

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.

You must modify the index in to your existing **pingfederate_oauth_clients_ext** table.

While there is no alter-table script provided, you can derive the setup from the new table-setup scripts, `oauth-client-management-<databaseServer>.sql`, found in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/sql-scripts` directory. Consult with your database administrator, as needed.

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 more recent) 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.

Alternatively, if you prefer to upgrade the older JDBC drivers at a later time, use the following steps to revert the JDBC connection pool library to BoneCP:

1. Edit `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.jdbc.DataSourceDeployerFactory.xml`.
2. Replace `dbcp` with `bonecp`.
3. Save the file.
4. Restart PingFederate.

   If you have a clustered PingFederate environment:

   a. Perform the steps above on the console node.
   b. Sign on to the PingFederate administrative console.
   c. Go to the **System**# **Cluster Management** screen.
   d. Click **Replicate Configuration** to push this change to all engine nodes.

ⓘ **Important:** Support for BoneCP as the JDBC connection pool library has been deprecated and will be removed in a future release.

ⓘ **Note:** PingFederate has been tested with vendor-specific JDBC 4.2 drivers. For more information, see . To obtain your database driver JAR file, contact your database vendor. Database driver file should be installed to the `<pf_install>/pingfederate/server/default/lib` directory. You must restart the server after installing the driver.

Changes in the database tables for log messages

Applicable only to customers who have configured Log4j or Log4j 2 to write log messages to database tables on Microsoft SQL Server.

For performance reasons, PingFederate 8.0 started using the `NVARCHAR` data type instead of the `VARCHAR` data type. The table-setup scripts (targeted for Microsoft SQL Server) in the `<pf_install>/pingfederate/server/default/conf/log4j/sql-scripts` directory have been updated. If you are upgrading from PingFederate 7.3 or an earlier version, consider updating the data type in the applicable tables accordingly for performance reasons.

Changes in the database table for account linking

Applicable only to customers who have created a database table for account linking on Microsoft SQL Server.

For columns in the **pingfederate_account_link** table using the `VARCHAR` data type, PingFederate 7.3 updated the data type to `NVARCHAR` for performance reasons. The table-setup script, `account-linking-sqlserver.sql`, in the `<pf_install>/pingfederate/server/default/conf/account-linking/sql-scripts` directory has been updated. If you are upgrading from PingFederate 7.2 R2 or an earlier version, consider updating the data type in the **pingfederate_account_link** table accordingly for performance reasons. Consult with your database administrator, as needed.

Changes in the database tables for OAuth clients

Applicable only to customers who have deployed Microsoft SQL Server to store their OAuth clients.

For columns in the **pingfederate_oauth_clients** and **pingfederate_oauth_clients_ext** tables using the `VARCHAR` data type, PingFederate 7.3 updated the data type to `NVARCHAR` for performance reasons. The table-setup script, `oauth-client-management-sqlserver`, in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/sql-scripts` directory has been updated. If you are upgrading from PingFederate 7.2 R2 or an earlier version with OAuth use cases, consider updating the data type in both tables accordingly for performance reasons. Consult with your database administrator, as needed.

Changes in the database tables for OAuth persistent grants and extended attributes

Applicable only to customers who have deployed Microsoft SQL Server to host their OAuth grant data store.

For columns in the **pingfederate_access_grant** and **pingfederate_access_grant_attr** tables using the `VARCHAR` data type, PingFederate 7.3 updated the data type to `NVARCHAR` for performance reasons. The table-setup scripts (targeted for Microsoft SQL Server) in the `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts` directory have been updated. If you are upgrading from PingFederate 7.2 R2 or an earlier version with OAuth use cases, consider updating the data type in both tables accordingly to avoid potential issues caused by incompatible data types between these two tables. Consult with your database administrator, as needed.

A new database table for OAuth persistent grant extended attributes

PingFederate 7.2 R2 introduced the capability to map attributes from initial authentication context to an access token attribute contract, which requires an update in the OAuth grant data store. If you are upgrading from PingFederate 7.2.1 or an earlier version with OAuth use cases, run the table-setup script, `access-grant-attribute-<database>.sql`, found in the `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts` directory for your database server. Consult with your database administrator, as needed.

New indexes in the database table for OAuth persistent grants

Since PingFederate 7.1 R3, a couple of indexes have been added to the **pingfederate_access_grant** table.

PingFederate version	Column	Index
7.3	`expires`	`EXPIRESIDX`
7.1 R3	`client_id`	`CLIENTIDIDX`

If you are upgrading from PingFederate 6.5 through 7.1.4, you must add both indexes to your existing **pingfederate_access_grant** table.

If you are upgrading from PingFederate 7.1 R3, 7.2.x, or 7.2 R2, you must add the `EXPIRESIDX` index for the `expires` column.

While there is no alter-table script provided, you can derive the setup from the new table-setup scripts, `access-grant-<databaseServer>.sql`, found in the `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts` directory. Consult with your database administrator, as needed.

Changes in a database table supporting nested group membership

Outbound provisioning of groups and nested group membership requires an update in the internal data store.

If you are upgrading from PingFederate 8.0.x, no action is required.

If you are upgrading from PingFederate 7.1.x through 7.3, consider running the alter-table script, `add-subgroup-membership-<database>.sql`, found in the `<pf_install>/pingfederate/server/default/conf/provisioner/sql-scripts/updates` directory for your database server to update the existing **group_membership** table. Consult with your database administrator, as needed.

If you are upgrading from PingFederate 7.0.1 or an earlier version, consider using the table-setup script, `provisioner-<database>.sql`, found in the `<pf_install>/pingfederate/server/default/conf/provisioner/sql-scripts` directory for your database server as a reference to add the new **group_membership** table to your existing internal data store.

Alternatively, you may create a new database using the table-setup script and let the outbound provisioner repopulate the new internal data store on its next run, regardless of the current version of your PingFederate.

> **CAUTION:** The table-setup script removes the existing (outbound provisioning) tables. If you wish to keep a copy of the current data, use the tools available from your database vendor to make a backup before running the table-setup script.

Consult with your database administrator, as needed.

Review log configuration

About this task

Starting with version 8.0, PingFederate uses Log4j 2 to write log messages. This upgrade improves performance and allows real-time log level adjustments. If you have not modified the Log4j configuration file (`log4j.xml`) in an earlier version, PingFederate 8.0 starts using Log4j2 after the upgrade process.

If the configuration file for Log4j 2 (or Log4j) has been modified in the source installation, refer to one of the subsequent topic for upgrade instructions.

Upgrading from PingFederate 8.x or 9.x

About this task

If the Upgrade Utility, the installer for Windows, or the install script for Red Hat Enterprise Linux determines that the Log4j2 configuration file (`log4j2.xml` in the `<pf_install>/pingfederate/server/default/conf` directory) has changed since it was originally installed, new features are *not* activated; the upgrade tools do not currently support automatic merging of customizations made to the existing logging configuration. Instead, these upgrade tools copy the modified `log4j2.xml` to the new installation intact and rename the configuration file from the product ZIP file using the new PingFederate version number. Both configuration files are located in the same `conf` directory.

To activate new features:

Steps

1. Review the new features by comparing the renamed Log4j2 configuration file against `log4j2.xml`.
2. Modify `log4j2.xml` to suit your needs.
3. If you have a clustered PingFederate environment, repeat step 2 for all applicable PingFederate nodes in the cluster.

Upgrading from PingFederate 6.x or 7.x

About this task

PingFederate 6.x and 7.x use Log4j to write log messages.

If the Upgrade Utility, the installer for Windows, or the install script for Red Hat Enterprise Linux determines that the Log4j configuration file (`log4j.xml` in the `<pf_install>/pingfederate/server/default/conf` directory) has changed since it was originally installed, these upgrade tools copy the modified `log4j.xml` to the new installation with a new name and installs the new `log4j2.xml` from the product ZIP file.

The new name for the previously customized `log4j.xml` is `log4j-old-<SourceVersion>.xml`, where `<SourceVersion>` is the version number of the source PingFederate installation.

Both configuration files are located in the `conf` directory.

To migrate custom changes from Log4j to Log4j 2:

Steps

1. Review custom changes by comparing `log4j-old-<SourceVersion>.xml` against `log4j.xml` that is located in the `pf-upgrade-<NewVersion>/reference-files/<SourceVersion>/server/default/conf` directory from the upgrade tool.

   Important: The `<SourceVersion>` values must match each other.

2. Modify `log4j2.xml` to suit your needs.

   Tip: The configuration syntax between Log4j and Log4j 2 varies. For more information, consult *Log4j 2 documentation* (logging.apache.org/log4j/2.x/manual/index.html).

3. If you are writing log messages to a database server, enter the database information into `log4j2.db.properties` in the same `conf` directory.
4. If you have a clustered PingFederate environment, repeat steps 2 and 3 for all applicable PingFederate nodes in the cluster.

Migrate other components

Some custom and integrated components may require additional steps after upgrade. For more information, Refer to subsequent topics.

Update custom authentication selector

Through the use of the PingFederate SDK, you may create a custom authentication selector by implementing the `AuthenticationSelector` interface. Most implementation returns `AuthenticationSelectorContext.ResultType.CONTEXT` as the result type, which requires no further action after an upgrade.

If your implementation returns `AuthenticationSelectorContext.ResultType.ADAPTER_ID` (an IdP adapter instance ID) or `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`. Furthermore, 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, refer to the Javadoc for the `AuthenticationSelector` interface and the `AuthenticationSelectorDescriptor` class.

Tip: The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.

Migrate to the integrated LDAP Username PCV

About this task

As of PingFederate 7.3, the integrated LDAP Username Password Credential Validator (PCV) is capable of returning 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**, it is recommended to migrate to the integrated LDAP Username PCV.

Steps

1. Create an instance of the integrated LDAP Username PCV.
   a. On the **System# Password Credential Validators** screen, click **Create New Instance**.
   b. On the **Type** screen, enter the required information and select **LDAP Username Password Credential Validator** from the list.
   c. On the **Instance Configuration** screen, select an LDAP data store from the list, enter a search base and a search filter, and select the scope of the search.

      ⓘ **Tip:** You may reuse the information from the existing **LDAP PCV with Extended Attributes** instance.

   d. On the **Extended Contract** screen, enter memberOf under **Extend the Contract** and click **Add.**

      Other commonly used attributes are DN, givenName, mail, and username.
   e. On the **Summary** screen, review the setup and click **Done**.
   f. On the **Manage Credential Validator Instances** screen, click **Save**.
2. In 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. When the **LDAP PCV with Extended Attributes** instance is no longer in-use, delete it from the **System# Password Credential Validators** screen.
4. Remove the `LDAPExtendedAttributesPCV-<version>.jar` file from the `<pf_install>/pingfederate/server/default/deploy` directory on all PingFederate servers.
5. Restart PingFederate on all PingFederate servers.

Migrate 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. While the integrated Username Token Processor and the deprecated Username Token Translator may be simultaneously deployed, it is recommended to migrate to the new token processor.

Steps

1. Go to the **Identity Provider# Token Processors** screen.
2. Click **Create New Instance** to create an instance of the integrated Username Token Processor.

   ⓘ **Important:** In the **Type** screen, select **Username Token Processor** from the list.

   ⓘ **Tip:** If you have multiple WS-Trust STS SP connections, you may 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** screen.

   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 in the **IdP Token Processor Mapping** screen.
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 in the **Identity Provider# Token Processors** screen.
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.

Reset files and variable for HSM

If your PingFederate installation is configured in a clustered environment with nCipher 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.

Verify the new installation

Integration kits and custom solutions may introduce third-party dependencies that could trigger runtime errors. We recommend that you perform runtime tests, verifying that the previously deployed use cases, including any OGNL expressions, are functional in the new installation.

PingFederate Performance Tuning Guide

The default configuration for PingFederate 9.3 is acceptable for most small size deployments; however, additional tuning is recommended for mission-critical and high-transaction volume deployments. Fine-tuning a few simple application and system level settings enables PingFederate to achieve maximum performance out of the hardware chosen for your deployment.

This topic addresses several areas of tuning such as logging, concurrency, memory, and Java-specific tuning options. The topic 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.

Note: An additional document related to performance, the *PingFederate Capacity Planning Guide*, is also available to customers as a performance data reference. This document is available from *support.pingidentity.com*.

Logging

Logging is a feature for tracking various aspects of the system's operation. It requires a certain amount of system resources, which affects the system's overall performance. Of particular expense is the actual writing to the log files. PingFederate 8.x uses the high-performance asynchronous logger from Log4j 2 to minimize the performance impact of logging runtime and administrative events, including status and error messages that can be used for troubleshooting. Audit information is logged synchronously to preserve transactional integrity.

Although the bulk of logging is executed asynchronously, decreasing the amount of information written to log files always provides the best possible performance.

Starting with version 8.2, PingFederate only records messages that are tagged with log level `INFO`, `WARN`, `ERROR`, and `FATAL` to the server log (and the provisioner log). Messages that are tagged `DEBUG` (or `TRACE`) are not recorded to optimize performance. Console logging is also disabled for the same reason.

For troubleshooting purpose, you may adjust the log level to `DEBUG` in the `log4j2.xml` file and optionally re-enable console logging.

Important: When debug messages and console logging are no longer required, ensure they are turned off.

Operating system tuning

This section contains tuning recommendations for your operating system. The tuning recommendations provided here are particularly useful in preventing deployment issues in high capacity environments.

Linux tuning

Implement these recommendations in your Linux environment to prevent deployment issues, increase the performance and the capacity of the networking stack (particularly TCP) and the file descriptor usage, and thus enabling PingFederate to handle a high volume of concurrent requests.

Network/TCP tuning

Add or modify the following entries in the `/etc/sysctl.conf` file:

```
##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:

```

            pf_user soft nofile 10400
pf_user hard nofile 10400
```

where *pf_user* is the user account used to run the PingFederate java process (or `*` for all user accounts).

Windows tuning

Implement these recommendations in your Windows environment to prevent deployment issues, increase the performance and the capacity of the networking stack (specifically the TCP socket), and thus enabling PingFederate to handle a high volume of concurrent requests.

Increase the number of available ephemeral ports

1. Start the Command Prompt application (`cmd.exe`).
2. Type `netsh int ipv4 show dynamicportrange tcp` to view the ephemeral ports.
3. Type `netsh int ipv4 set dynamicport tcp start=1025 num=64510` to increase the range of the ephemeral ports. (Administrative privileges required.)
4. Reboot the server.
5. Type `netsh int ipv4 show dynamicportrange tcp` to confirm the updated port range.

Reduce socket TIME_WAIT delay

1. Start the Registry Editor application (`regedit.exe`).
2. Navigate 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

Generally speaking, the more requests that are processed in parallel, the higher the number of requests that are 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

It is important to note that this topic serves as a guideline for optimizing the concurrency of your deployment. On a large system with multiple CPU (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 (reasonably expected) user load. This way, CPU resources are not under-utilized while still allowing room for occasional load spikes. The level of concurrency in PingFederate may 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 topic therefore serves as a guideline of where to start when tuning the server.

Acceptor queue size

About this task

For optimal performance, particularly in larger deployments, PingFederate uses a non-blocking I/O model to process requests since version 8.0.1. As needed, administrators may fine-tune the queue size parameter, acceptQueueSize.

Steps

1. Stop PingFederate.
2. Edit the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file.

   Consider making a backup copy of this file.
3. Find the following section:

```
<Call id="httpsPrimaryConnector" name="addConnector">
  <Arg>
    <New
 class="com.pingidentity.appserver.jetty.server.connector.ServerConnector">
      <Arg name="server"><Ref refid="RuntimeServer" /></Arg>
      <Arg name="acceptors" type="int"><Property name="ssl.acceptors"
 default="0"/></Arg>
      ...
      <Set name="acceptQueueSize">512</Set>
    </New>
  </Arg>
</Call>
```

4. If the queue reaches its maximum size, additional requests will receive a connection (refused) error. If this occurs in your environment, you can increase the value of the acceptQueueSize element.
5. When finished, save your changes and restart PingFederate.
6. For a clustered PingFederate environment, repeat these steps on each engine node as needed.

Server thread pool

About this task

When tuning the server thread pool, it is best to size the system based on expected user load, setting 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 to between 25% and 50% higher than the minimum to handle load spikes. Testing has shown that PingFederate performs quite well when the server thread pool is sized between 25 and 50 server threads per available CPU core, assuming sufficient memory (see *Memory* on page 977). Note that this is guidance and should not necessarily be scaled directly on larger systems.

For example, if you are running PingFederate on a system with 24 CPU cores, it does not 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

Steps

1. Stop PingFederate.
2. Edit the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file.

   Consider making a backup copy of this file.
3. Find the following section:

   ```
   <Get name="ThreadPool">
       <Set name="minThreads" type="int">10</Set>
       <Set name="maxThreads" type="int">200</Set>
       <Set name="detailedDump">false</Set>
   </Get>
   ```
4. Modify it using the following guidelines:
   - Set minThreads to (available CPU cores * 25)
   - Set maxThreads to (available CPU cores * 50)

   For example, if your PingFederate system has 1 CPU with 4 cores, the total available cores is 4. The configuration would be:

   ```
   <Get name="ThreadPool">
       <Set name="minThreads" type="int">100</Set>
       <Set name="maxThreads" type="int">200</Set>
       <Set name="detailedDump">false</Set>
   </Get>
   ```
5. When finished, save your changes and restart PingFederate.
6. For a clustered PingFederate environment, repeat these steps on each engine node as needed.

Connection pools to data stores

About this task

Database (JDBC) and LDAP data stores use connection pooling to improve the performance and efficiency of communicating with external systems. Connection pools improve efficiency by maintaining persistent connections to the database or LDAP server thus preventing the expense of creating the connection on demand. Connection pools also allow more control over the load placed on the back-end server.

Much like the server thread pool, for optimal performance, a number of connections is required to handle most (if not all) the requests in parallel. It may 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 database and LDAP resources. For optimal performance, connection pools should be sized large enough to handle between 50% and 100% of the number of concurrent requests the server is expected to encounter most often.

Of course, it is also very important to understand the capacity and limitation of the database or LDAP server and to size the connection pool accordingly. Sizing the connection pool beyond the capability of the back-end server could lead to PingFederate flooding the data store without any performance improvement.

Steps

1. For JDBC data stores:
   a. On the **System**# **Data Stores** screen, select the applicable JDBC data store.
   b. On the **Database Config** screen, click **Advanced**.
   c. On the **Advanced Database Options** screen:
      - Set the **Minimum Pool Size** value to 50% of the maxThreads value.
      - 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.

   Tip: The maxThreads value is defined in the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file (see *Server thread pool* on page 975).
2. For LDAP data stores:
   a. On the **System**# **Data Stores** screen, select the applicable LDAP data store.
   b. On the **LDAP Configuration** screen, click **Advanced**.
   c. On the **Advanced LDAP Options** screen:
      - Set the **Minimum Connections** value to 50% of the maxThreads value.
      - Set the **Maximum Connections** value to between 75% and 100% of the maxThreads value, subject to the capability of the back-end database server.

   Tip: The maxThreads value is defined in the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file (see *Server thread pool* on page 975).

   Note: As of PingFederate 7.2, separate connection pools are created for bind requests and search requests. This dramatically improves performance of LDAP authentications, especially when using LDAPS, because bind requests no longer require unique connections. Each connection pool (bind and search) is configured individually per the values set for minimum and maximum connections
3. For a clustered PingFederate environment, replicate the changes to all engine nodes on the **System**# **Cluster Management** screen.

Memory

After the CPU, memory is the most important resource in your deployment. The previous section describes how to configure PingFederate to support more concurrent requests. With increased concurrency comes an increase in the memory required. Moreover, PingFederate is a Java application, and although this document is not to be considered a guide to garbage collection theory or ergonomics, consideration must be given to tuning that affects, or is affected by, garbage collection.

JVM heap

The most important tuning that can be applied to the Java Virtual Machine (JVM) is the size of the heap memory. 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 expensive processes, and thus detrimental to performance. Sizing the heap to ensure an adequate amount of memory is available but still manageable to garbage collection is important in optimizing overall performance.

PingFederate attempts to optimize JVM heap and garbage collector settings based on available system resources at the time of installation and upgrade. Depending on your environment, you may override these settings at a later time.

Additional considerations

The JVM can grow the heap from the value of the minimum heap variable to the value of the maximum heap variable. However, growing the heap is not free. Requesting memory from the operating system is often an expensive exercise. In addition, the JVM must also reorganize the heap to account for the memory being added. If conserving memory in your deployment is important, then setting a lower value for the minimum heap than that of the maximum heap ensures that you are not reserving memory that you are not specifically using. If, however, you have enough memory such that a certain amount is easily earmarked for the PingFederate server, we recommend that you adjust the size of the heap by setting 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 being used exceeds the value of minimum heap.

Garbage collectors

The JVM comes with a wide variety of tuning options. Although the virtual machine should configure itself to run the best it can in most situations; it is sometimes advantageous to configure it for a specific application. This section explores a few additional options that can be applied to potentially improve the operation of PingFederate in your deployment. In all cases, it is recommended that you test these options to ensure that they do benefit your deployment before enabling them in a production environment.

Parallel collector

By default, the *server HotSpot* JVM selects the parallel collector on machines with multiple CPUs (or cores). On machines with a single CPU, the serial collector is used.

On machines with eight or more hardware threads, the parallel collector uses a fraction of them as the number of garbage collector threads; the fraction is roughly 5/8. If the number of hardware threads drops below 8, the number of garbage collector threads matches the number of hardware threads.

The number of garbage collector threads can be overridden by the `-XX:ParallelGCThreads=`*n* JVM option, where *n* is the desired number of garbage collector threads. It is generally recommended to leave `ParallelGCThreads` as the default. Specifically, setting to a value greater than the number of CPUs will not improve garbage collection performance as the GC threads will all contend for CPU time causing the operating system to context switch between them. Setting to less than the number of CPUs can cause longer than necessary pause times because not all available CPU resources will be utilized.

The parallel collection is generational, so minor (young generation only) and major (entire heap) collections do occur. Because PingFederate uses a much larger proportion of short to medium lived objects, consider applying a young generation bias to the JVM heap, which will improve performance because the parallel scavenge copying collector used for the young generation are quite fast (see *Young generation bias* on page 979).

Concurrent mark sweep collector

The concurrent mark sweep (CMS) collector is suitable for applications that prefer shorter garbage collection pauses, can afford to share CPU resources with the garbage collector, have a large set of long-lived objects in the tenured (or old) generation, and run on machines with multiple CPUs.

The CMS collection is generational, so minor (young generation only) and major (entire heap) collections occur. Because PingFederate uses a much larger proportion of short to medium lived objects, the CMS collector is not generally the best fit. However, if you intend on using a JVM heap greater than 6 GB on larger machines with eight or more CPUs, the CMS collector may provide shorter pause time than the parallel collector in major collections. Applying a young generation bias to the JVM heap will improve performance when using the CMS collector because it employs the same parallel scavenge copying collector for the young generation as the parallel collector.

As needed, the CMS collector can be enabled by the `-XX:+UseConcMarkSweepGC` JVM option.

Garbage first collector

The garbage first (G1) collector is best for machines with multiple CPUs and a JVM heap of 6 GB or more. The G1 collector is designed to achieve high throughput while meeting its pause times goal (for garbage collection).

If you intend to use the G1 collector, it is strongly advised that you remove any sizing options specific to the young generation. The reason is that the G1 collector self-tunes by adjusting the size and nature of the various heap regions to meet the pause time goal. By setting a fixed amount of memory to be used for young generation regions, it limits its self-tuning options.

When applicable, the G1 collector can be enabled by the `#XX:+UseG1GC` JVM option.

ⓘ **Tip:** For additional information about each garbage collector, please refer to *Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide* from Oracle (docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/).

Young generation bias

Without getting into too much detail on the generational memory management model in Java, the basic principal is that new objects are created in the *young generation* and are garbage collected when they are no longer used. If an object is created, and a reference is maintained, that object is eventually moved to the *old generation*.

ⓘ **Note:** If you intend to use the garbage first (G1) collector, let the JVM handles this aspect of the heap because specific settings can affect the performance of the G1 collector adversely.

Skip to *Fine-tune JVM options* on page 985 for instructions on fine-tuning memory and garbage collection settings.

The processing model of PingFederate is mostly geared towards short-lived transactions (single sign-on and token processing) and not long held, interactive, user sessions. As such, most of the objects that are created are relatively short-lived. It does not make sense to *promote* short-lived objects to the old generation because they are probably not going to be needed for long anyway. The problem is that when the young generation fills up, space must be made for new objects. So objects that are in the young generation and still in use must be moved to the old generation, which has a cost. Moreover, those objects will eventually become garbage and need to be collected from the old generation.

The old generation is typically more expensive to clean up than the young generation because the old generation is cleaned only during a full garbage collection (meaning that the JVM has almost reached the value of the maximum heap variable (-Xmx) and the entire heap must be cleaned). However, the young generation is garbage collected more frequently and with multiple threads by default (on systems with multiple cores), so the pauses for collections are shorter.

By default, the JVM tends to size the generations biased to the old generation, giving it most of the total space of the heap. This means more frequently moving objects from the young generation into the old generation to make space for new objects, and more frequent *full collections* as the old generation fills up. By configuring the JVM to provide more memory to the young generation, the frequency of the more costly full collections is reduced. You can either specify fixed values for the size of the young generation or modify the ratio of young generation to old generation.

To specify a fixed value for the young generation, use the `–XX:NewSize=` and `–XX:MaxNewSize=` arguments. These arguments are to the young generation what the minimum heap variable (-Xms) and the maximum heap variable (-Xmx) are to the entire heap: the `–XX:NewSize=` and `–XX:MaxNewSize=` arguments define the initial (or minimum) and the maximum sizes of the young generation. The same reasoning that applies to the minimum and maximum heap variables are also applicable when adjusting these values.

To specify a ratio between the old and new generation size, use the `-XX:NewRatio=` argument. For example, setting `-XX:NewRatio=3` means that the ratio between the young and old generation is 1:3. Put another way, the size of the young generation is one fourth of the total heap size.

In a mostly short-lived object environment, it is recommended that 50-60% of the heap be given to the young generation; for example:

Young generation bias condition	JVM options

To fix a heap size of 2 GB with 50% the young generation bias using the `NewSize` argument	`-Xms2048m`
	`-Xmx-2048m`
	`-XX:NewSize=1024m`
	`-XX:MaxNewSize=1024m`

To fix a heap size of 2 GB with 50% the young generation bias using the `NewRatio` argument	`-Xms2048m`
	`-Xmx-2048m`
	`-XX:NewRatio=1`

The **`memoryoptions`** utility

PingFederate installation and upgrade tools use its **`memoryoptions`** utility to detect the available resources at the time of the installation or upgrade and record the recommended options for Java heap and the garbage collector in a configuration file. As needed, administrators may 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.

Installation and upgrade

When executed by a platform-specific installer or a subsequent rerun of the utility, the **`memoryoptions`** utility creates a backup copy of the current `jvm-memory.options` file (if any), detects available system resources at the time, and records the recommended 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.

When executed by the upgrade tools, depending the selected tool and whether the `jvm-memory.options` exists in the source installation, the expected behavior differs. Generally speaking, the `jvm-memory.options` file from the source installation is preserved without new recommended values.

`memoryoptions` and installation

The PingFederate platform-specific installer runs the **`memoryoptions`** utility in an attempts to optimize JVM heap and garbage collector options based on available system resources at the time of installation. As needed, administrators may re-run the utility or manually edit these options at a later time.

When executed by a platform-specific installer or a subsequent rerun of the utility, the **`memoryoptions`** utility creates a backup copy of the current `jvm-memory.options` file (if any), detects available system resources at the time, and records the recommended 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. Refer to the following table for information regarding expected behaviors.

Installation medium	Expected behavior
PingFederate installer for Windows *or* PingFederate install script for Red Hat Enterprise Linux	▪ The installer or the install script creates a new PingFederate installation. ▪ The installer or the install script runs the **`memoryoptions`** utility, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them in the `jvm-memory.options` file. ▪ Both tools configure PingFederate to run as a service. ▪ The recommended 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. ▪ Note that the default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, administrators should run the **`memoryoptions`** utility, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them in the `jvm-memory.options` file. ▪ The JVM options as a result of the execution of the **`memoryoptions`** utility (or a manual edit of the `jvm-memory.options` file) are activated as PingFederate restarts. **PingFederate as a server 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, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them 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 recommended options are activated as the PingFederate service starts.

`memoryoptions` and upgrade

When executed by the upgrade tools, depending the selected tool and whether the `jvm-memory.options` exists in the source installation, the expected behavior differs. Generally speaking, the `jvm-memory.options` file from the source installation is preserved without new recommended values. Refer to the following table for information regarding expected behaviors.

Upgrade path	**Expected behavior when the `jvm-memory.options` file does not exist in the source installation**
PingFederate installer for Windows *or* PingFederate install script for Red Hat Enterprise Linux	▪ The installer or the install script creates a new PingFederate installation. ▪ The installer or the install script runs the **`memoryoptions`** utility, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them in the `jvm-memory.options` file. ▪ Both tools configure PingFederate to run as a service. ▪ The recommended options are activated as the PingFederate service starts.
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 JVM options set in the default `jvm-memory.options` file are activated as PingFederate starts. ▪ Note that the default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, administrators should run the **`memoryoptions`** utility, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them in the `jvm-memory.options` file. ▪ The JVM options as a result of the execution of the **`memoryoptions`** utility (or a manual edit of the `jvm-memory.options` file) are activated as PingFederate restarts. **PingFederate as a server 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, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them 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 recommended options are activated as the PingFederate service starts.

Upgrade path	**Expected behavior when the `jvm-memory.options` file 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. ▪ Note that the default JVM options are conservative. For most deployment scenarios using various physical or virtual resources, administrators should run the **`memoryoptions`** utility, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them in the `jvm-memory.options` file. ▪ The JVM options as a result of the execution of the **`memoryoptions`** utility (or a manual edit of the `jvm-memory.options` file) are activated as PingFederate restarts.

Upgrade path	**Expected behavior when the `jvm-memory.options` file 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 in turn 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 JVM options are activated as the PingFederate service starts.
PingFederate install script for Red Hat Enterprise Linux *or* PingFederate Upgrade Utility (upgrade.sh)	▪ The install script or 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 the `jvm-memory.options` file 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, which is designed to determine the recommended Java heap and garbage collector options based on the available resources and to record them 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. To restore the *preserved* JVM options from the source installation, follow these steps: **1.** Rename the current `jvm-memory.options` file. For example: `jvm-memory.options.backup` **2.** Look for the preserved `jvm-memory.options` file. The preserved file was renamed with a time stamp. **3.** Remove the time stamp from the file name. At this point, 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`. 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. **7.** Restart the PingFederate Windows service. The preserved JVM options are activated as the PingFederate service starts.

Fine-tune JVM options

About this task

PingFederate reads 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. It is also recommended that a backup copy be made prior to any manual edits. Note that empty lines and comments (indicated by a leading `#` character) are ignored. Additionally, JVM options are not required to be organized in any specific order.

Steps

1. Edit the `<pf_install>/pingfederate/bin/jvm-memory.options` file.
2. To configure a specific heap size, edit the minimum (`-Xms`) and maximum (`-Xmx`) heap options.

   The valid unit qualifiers are `k` for kilobytes, `m` for megabytes, and `g` for gigabytes. In other words, `-Xmx1536m` and `-Xmx1.5g` are equivalent.

   For example, to fix the JVM heap size to 2 GB, configure the minimum and maximum options as follows:

   ```
   -Xms2g
   -Xmx2g
   ```
3. To override the number of garbage collection threads used by the parallel collector, add the `-XX:ParallelGCThreads=n` option, where *n* is the desired number of garbage collector threads.

   For example, to configure the parallel collector to use four threads, add to the configuration file the following option:

   ```
   -XX:ParallelGCThreads=4
   ```
4. To enable the concurrent mark sweep (CMS) collector, remove the `-XX:+UseParallelGC` option (or the options pertaining to another garbage collector) and replace it with the `-XX:+UseConcMarkSweepGC` option.
5. To enable the garbage first (G1) collector, remove the `-XX:+UseParallelGC` option (or the options pertaining to another garbage collector) and replace it with the `#XX:+UseG1GC` option.

   Note: If you enable the G1 collector, it is recommended that you remove any sizing options specific to the young generation. Skip to *this step*.
6. To configure young generation-specific sizing options, edit the minimum (`-XX:NewSize`) and maximum (`-XX:MaxNewSize`) options for the young generation space.

   For example, to fix the young generation bias to 1 GB, set the minimum and maximum options as follows:

   ```
   -XX:NewSize=1024m
   -XX:MaxNewSize=1024m
   ```
7. To remove young generation-specific sizing options completely, remove the aforementioned options or add a leading `#` character.
8. To add additional JVM options, insert the applicable options to the file.

   For example, to enable the aggressive options flag, configure the file as follows:

   ```
   ...

   # Enable the aggressive options flag
   ```

```
-XX:+AggressiveOpts
```

(The comment is optional.)

9. When finished, save your changes.
10. If PingFederate is configured to run as a service on a Windows server, follow these steps:
    a. Open a command prompt and go to the `<pf_install>/pingfederate/sbin/wrapper` directory.
    b. Run `generate-wrapper-jvm-options.bat`.

       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.
    c. Close the command prompt.
11. Restart PingFederate.
12. For a clustered PingFederate environment, repeat these steps on each engine node as needed.

Hardware security modules

For optimal security, PingFederate can be configured to use a hardware security module (HSM) for cryptographic material storage and operations. Standards such as the Federal Information Processing Standard (FIPS) 140-2 require the storage and processing of all keys and certificates on a certified cryptographic module.

(For more information, see *Supported hardware security modules* on page 90.)

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. It is recommended that you 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

For deployments with hundreds of connections or OAuth clients (or both), if noticeable delays are observed 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.

References

Memory management

Java Platform, Standard Edition HotSpot Virtual Machine Garbage Collection Tuning Guide

(docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/)

Hotspot JVM arguments

Java HotSpot VM Options (www.oracle.com/technetwork/java/javase/tech/vmoptions-jsp-140102.html)

Legal Information

Index