PingFederate®

Server 9.1.4

Contents

Copyright

PingFederate Server 9.1.4
November 2018

Ping Identity Corporation
1001 17th Street, Suite 100
Denver, CO 80202
U.S.A.

Trademarks

Disclaimer

Document lifetime

Ping Identity may occasionally update online documentation between releases of the related software. Consequently, if this PDF was not downloaded recently, it may not contain the most up-to-date information. Please refer to the *PingFederate online documentation* website for the most current information.

From the web site, you may also download and refresh this PDF if it has been updated, as indicated by a change on this date: **November 15, 2018**.

Get Started with PingFederate Server 9.1.4

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.

This guide consists of:

- *Introduction to PingFederate* on page 9 — A high-level view of federated identity, secure web SSO, and PingFederate features.
- *Installation* on page 12 — How to install PingFederate and run the administrative console for the first time.
- *PingFederate administrative console* on page 31 — A primer on using the administrative console and configuration screens.
- *Supported standards* on page 33 — An overview of industry standards that PingFederate supports, including the Security Assertion Markup Language (SAML) and WS-Federation.
- *Supported hardware security modules* on page 56 — How to install and configure PingFederate with a supported HSM as part of compliance with the Federal Information Processing Standard (FIPS) 140-2.

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 33.

Service providers and identity providers

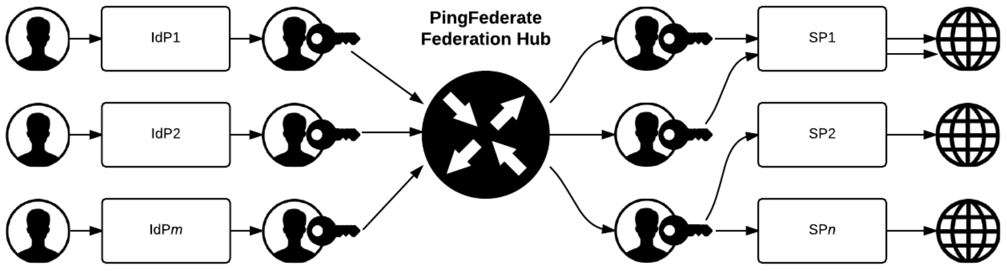
Identity federation standards identify two operational roles in an SSO transaction: the *identity provider* (IdP) and the *service provider* (SP). An IdP, for example, might be an enterprise that manages accounts for a large number of users who may need secure access to the Web-based applications or 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.

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 50).

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 62).

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. Fore more information, see *About OAuth* on page 65.

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 84.

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 85).

Additional features

PingFederate lightweight, stand-alone 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.

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 72.

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 63.

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 85). 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 is Ping Identity's multi-tenant, identity-as-a-service (IDaaS) solution. PingOne® enables browser-based SSO and user provisioning for Identity Providers, and provides application providers with a rapid-deployment SSO capability. PingOne can be used together with PingFederate to provide 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*.

Installation

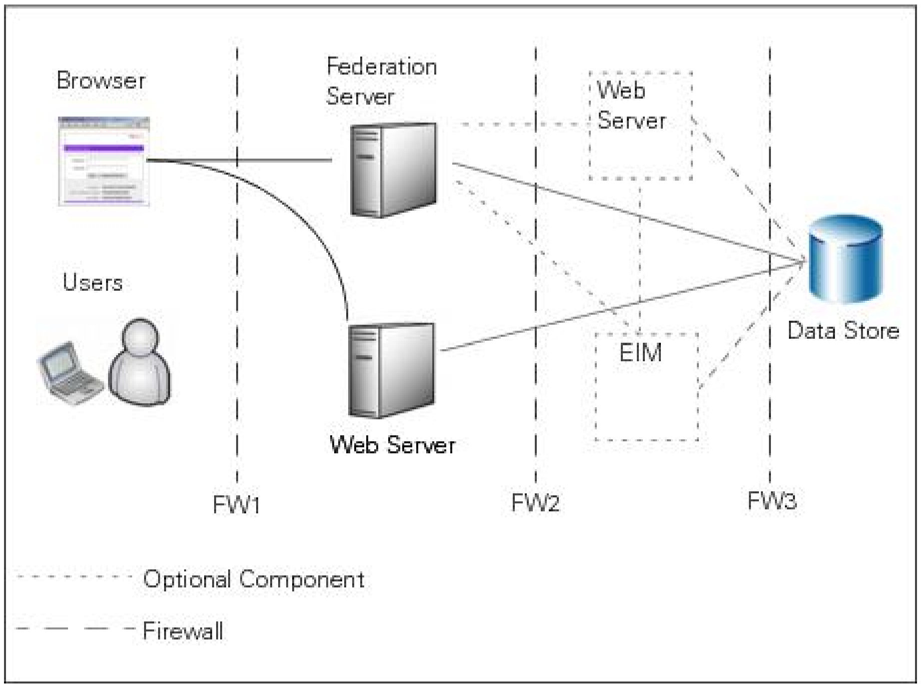
PingFederate is packaged as a stand-alone server based on J2EE application server technology. A new installation involves the following tasks:

- *Deployment options* on page 12
- *System requirements* on page 14
- *Port requirements* on page 17
- *Install Java* on page 19
- *Installation options* on page 20
- *Initial Setup wizard* on page 24

Deployment options

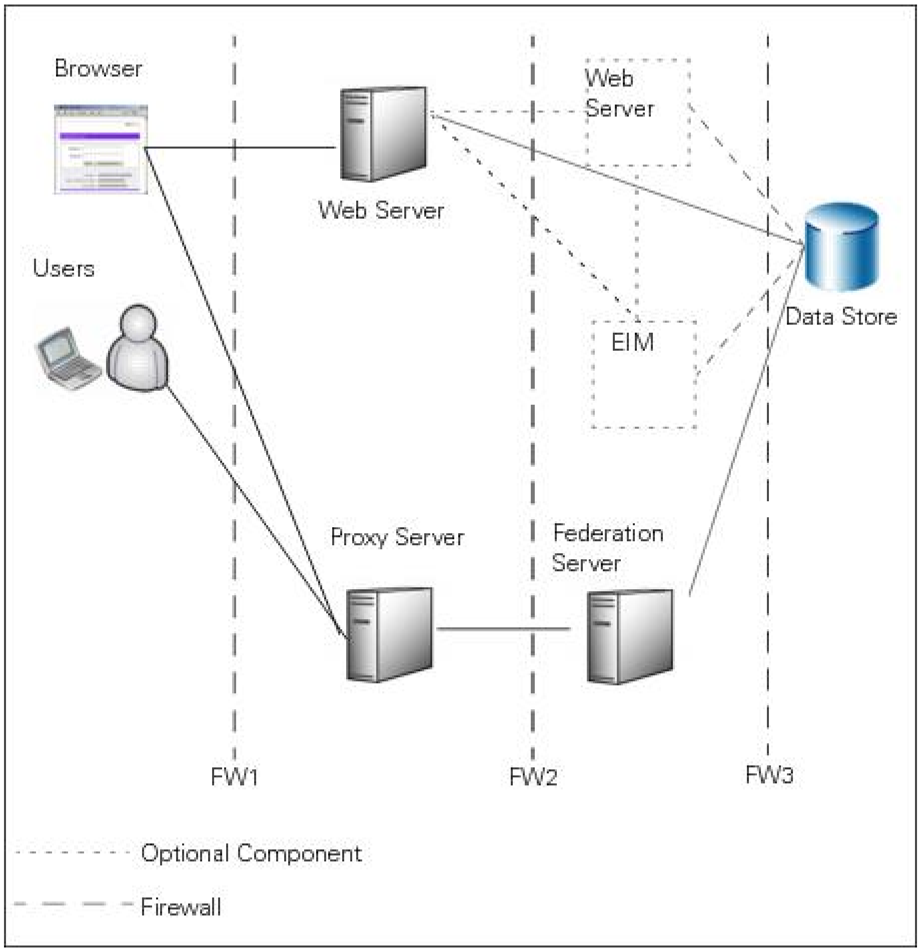
There are many options for deploying PingFederate in your network environment, depending on your needs and infrastructure capabilities.

For example, you can choose a stand-alone 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

PingFederate is certified as compatible for deployment and configuration with the minimum system specifications defined below.

Software 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

- Canonical Ubuntu 16.04 LTS
- Canonical Ubuntu 18.04 LTS
- Microsoft Windows Server 2012 Standard
- Microsoft Windows Server 2012 R2 Datacenter
- Microsoft Windows Server 2016
- Oracle Enterprise Linux 6.9 (Red Hat compatible kernel)
- Oracle Enterprise Linux 7.5 (Red Hat compatible kernel)

- Oracle Solaris 10
- Red Hat Enterprise Linux ES 6.9
- Red Hat Enterprise Linux ES 7.5
- SUSE Linux Enterprise 11 SP4
- SUSE Linux Enterprise 12 SP3

Docker support

- Docker version: 18.03.1-CE
- Host operating system: Canonical Ubuntu 16.04 LTS
- Kernel: 4.4.0-1052-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

Oracle Java SE Runtime Environment (Server JRE) 8 update 172

Browsers

Runtime server

- Chrome
- Edge
- Firefox
- Internet Explorer 11
- Safari
- Android 8 (Chrome)
- iOS 11 (Safari)

Administrative server

- Chrome
- Firefox
- Internet Explorer 11

TLS protocol

Runtime server and administrative server

- TLS 1.2

Data store integration

User-attribute lookup

- PingDirectory™ 6.0, 6.1, 6.2, and 7.0
- Microsoft Active Directory 2012 and 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2014 and 2016
- Oracle Database 12c

- Oracle MySQL 5.7
- PostgreSQL 9.6.1
- Custom implementation through the PingFederate SDK

SaaS or SCIM outbound provisioning

Provisioning channel data source

- PingDirectory 6.0, 6.1, 6.2, and 7.0
- Microsoft Active Directory 2012 and 2016
- Oracle Directory Server Enterprise Edition 11g

Provisioning internal data store

- Microsoft SQL Server 2014 and 2016
- Oracle Database 12c
- Oracle MySQL 5.7
- PostgreSQL 9.6.1

SCIM inbound provisioning

- Microsoft Active Directory 2012 and 2016
- Custom implementation through the PingFederate SDK

Just-in-time (JIT) inbound provisioning

- Microsoft SQL Server 2014 and 2016

Account linking

- PingDirectory 6.0, 6.1, 6.2, and 7.0
- Microsoft Active Directory 2012 and 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2014 and 2016
- Oracle Database 12c
- Oracle MySQL 5.7
- PostgreSQL 9.6.1

OAuth client configuration

- PingDirectory 6.0, 6.1, 6.2, and 7.0
- Microsoft Active Directory 2012 and 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2014 and 2016
- Oracle Database 12c
- Oracle MySQL 5.7
- PostgreSQL 9.6.1
- Custom implementation through the PingFederate SDK

OAuth persistent grants

- PingDirectory 6.0, 6.1, 6.2, and 7.0
- Microsoft Active Directory 2012 and 2016
- Oracle Directory Server Enterprise Edition 11g
- Microsoft SQL Server 2014 and 2016
- Oracle Database 12c
- Oracle MySQL 5.7
- PostgreSQL 9.6.1
- Custom implementation through the PingFederate SDK

Registration and profile management of local identities

- PingDirectory 6.0, 6.1, 6.2, and 7.0

Note: PingFederate has been tested with vendor-specific JDBC 4.1 drivers. Contact your vendor for driver information.

Hardware security module (optional)

Gemalto SafeNet Luna SA

- Firmware version 6.2.1
- Client driver version: 5.3

Thales nShield Connect

- Firmware version: 12.40.0
- Client driver version: 12.40.2

Hardware requirements

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.

Minimum hardware requirements

- Intel Pentium 4, 1.8 GHz processor
- 1 GB of RAM
- 1 GB of available hard drive space

Minimum hardware recommendations

- Multi-core Intel Xeon processor or higher

  4 CPU/Cores recommended
- Multi-core SPARC processor (Solaris)

  4 CPU/Cores recommended
- 4 GB of RAM

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

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 (Type of Traffic)	Protocol, Direction, Transport, Default Port	Source	Destination	Description
Administrative console	HTTPS, inbound, TCP, 9999	Administrator browser, administrative API REST calls, web service calls to the Connection Management Service Applicable to the console node in a clustered PingFederate environment	Administrative node[1]	Used for incoming requests to the administrative console. Configurable in the `run.properties` file.
Runtime engine	HTTPS, inbound, TCP, 9031	Client browser; mobile devices; web service calls to the SSO Directory Service, the OAuth Client Management Service, and the OAuth Access Grant Management Service; Session Revocation API REST calls 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. [2]
Cluster traffic (TCP)	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 (TCP)	JGroups, inbound, TCP, 7700	PingFederate peer servers in a clustered	Administrative node and runtime engine nodes	Used by other nodes in the cluster as part of the cluster's failure-detection mechanism when the

[1] In a standalone environment, your PingFederate server is both the administrative node and the runtime engine node. In a clustered environment, you configure one of your Pingfederate as the sole administrative node in the cluster and the rest of the PingFederate servers as runtime engine nodes.

[2] The pf.secondary.https.port, if activated in the `run.properties` file, needs to be open as well.

Service (Type of Traffic)	Protocol, Direction, Transport, Default Port	Source	Destination	Description
		PingFederate environment		transport mode for cluster traffic is set to TCP (the default behavior). Configurable in the `run.properties` file.
Cluster traffic (TCP, optional)	JGroups, outbound, TCP, 443	PingFederate peer servers in a clustered PingFederate environment	Amazon Simple Storage Service (Amazon S3) or an OpenStack Swift server	Used by all nodes when the optional dynamic discovery mechanism is enabled.
Cluster traffic (UDP)	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® integration (optional)	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, monitoring of PingFederate from PingOne, authenticating end users against the PingOne directory.
Active Directory domains/ Kerberos realms (optional)	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.

Note: Depending on the integration kits deployed and the connecting third-party systems (such as email server or SMS service provider), you may need to open additional ports.

Install Java

You must install the Oracle Java SE Runtime Environment (Server JRE) before running PingFederate, see *System requirements* on page 14 for more information.

Tip: Due to import control restrictions, the standard Server JRE distribution supports strong but not unlimited encryption. Stronger encryption is optional in several PingFederate and plug-in configurations. To use the strongest encryption, when permissible, after installing the Server JRE, download and install the appropriate version of “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files” from the *Oracle Downloads* website (www.oracle.com/technetwork/java/javase/downloads/index.html).

1. Download and install the Server JRE from the *Oracle Downloads*.
2. Set the JAVA_HOME environment variable to the Server JRE 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

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

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 Server JRE is installed and the required environment variables are set correctly (see *Install Java* on page 19).
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 the PingFederate service on Windows manually* on page 27.

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 56.

Install PingFederate on Red Hat Enterprise Linux

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 Server JRE is installed and the required environment variables are set correctly (see *Install Java* on page 19).
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 the PingFederate service on Linux manually* on page 27.

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 56.

Install PingFederate on UNIX/Linux

Refer to *System requirements* on page 14 for a list of qualified UNIX and Linux operating systems.

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 Server JRE is installed and the required environment variables are set correctly (see *Install Java* on page 19).
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 the PingFederate service on Linux manually* on page 27.

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 56.

Start and stop PingFederate

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.

- 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.

Operating system	Application mode	Steps
	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.
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 the 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.

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.

Initial Setup wizard

The first time you run the PingFederate administrative console, the **Initial Setup** wizard guides you through the process of configuring your identity federation settings and optionally connecting PingFederate to PingOne® to deploy a powerful on-premise and cloud-based hybrid solution. The tasks include:

- *Connect PingFederate to PingOne* on page 24 (optional)
- *Review or import your license* on page 24
- *Enter the basic information* on page 24
- *Select your federation roles* on page 25
- *Configure identity provider settings* on page 25 (optional)
- *Create an administrator account* on page 27
- *Review the initial configuration* on page 27

Connect PingFederate to PingOne

PingOne is Ping Identity's multi-tenant, identity-as-a-service (IDaaS) solution. PingOne® enables browser-based SSO and user provisioning for Identity Providers, and provides application providers with a rapid-deployment SSO capability. PingOne can be used together with PingFederate to provide a powerful solution combining the benefits of an on-premise deployment with the flexibility of a cloud solution.

When you select to connect PingFederate to PingOne, the administrative console guides you to create a managed SP connection. Once 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. When PingFederate detects such changes, the administrative console prompts you decide whether to update PingOne or to disconnect from PingOne in a banner message. In addition, PingFederate automatically downloads new signing certificates from PingOne and updates the connection.

1. Click **Sign on to PingOne to get your activation key**.
2. Sign on using your PingOne admin portal credentials.
3. Follow the on-screen instructions to connect PingFederate as the identity repository in PingOne.
4. Copy the **Activation Key** value.
5. Close the browser tab and go back to the PingFederate administrative console.
6. On the **PingOne Account** screen, paste your activation key.
7. Click **Next**.

If you prefer to setup PingFederate without PingOne for now, click **Next** to continue. When you are ready to connect PingFederate to PingOne, go to the **Server Configuration** menu and click **Connect to PingOne**.

Review or import your license

On the **License** screen, configure and review your license information.

- If you have selected to connect PingFederate to PingOne®, the **Initial Setup** wizard automatically downloads a 30-day trial license after validating your activation key.

  If you wish to import your license file, you may do so at this point. Alternatively, you can import your license file on the **Server Configuration** > **License Management** screen *before* the trial license expires.
- If you have opted to setup PingFederate without PingOne, import your license file.

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

Click **Next** to continue.

Enter the basic information

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

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

Tip: 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 server.

2. Specify your **Entity ID**.

   Note: If you have selected to connect PingFederate to PingOne®, the **Entity ID** field is pre-populated for you based on your PingOne setup.
3. Click **Next**.

Select your federation roles

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

1. Select at least one role for your PingFederate server.

   Note: If you have selected to connect PingFederate to PingOne®, the **Identity Provider** role is activated for your convenience. You may select additional roles as needed.
2. Click **Next**.

Configure identity provider settings

The **Identity Provider Configuration** screen appears when the **Identity Provider** role is activated. Use this screen to:

- Connect your Active Directory as an LDAP data store
- Create adapters and authentication selectors to authenticate end users via the Kerberos protocol or a login form based on end-user browsers and your network topology
- Enable SSO from PingOne® to the PingFederate administrative console or create a local administrative account to access the console

- To continue with the **Initial Setup** wizard, click **Begin** or **Connect to Active Directory**.

  Note: The **Identity Provider Configuration** screen is also the second step in the *Connect to PingOne* configuration wizard from the **Server Configuration** menu; this is the use case where you decided not to connect PingFederate to PingOne in the past but would like to do so now.

Connect your Active Directory

As an identity provider, you often need to supply additional information about your users, such as their first and last names and email addresses, to single sign-on to the SPs including PingOne®. If Active Directory (AD) is your user repository, use the **Connection** screen to establish a secure connection to your AD LDAP server.

Based on the information provided, the **Initial Setup** wizard also creates an LDAP Username Password Credential Validator instance and an HTML Form Adapter instance automatically for you, such that your users can authenticate through a login form using their AD credentials.

1. Enter the hostname and the access credentials.
2. Specify a **Search Base**. This is the starting point in your AD where PingFederate looks for users and groups.
3. Modify the pre-populated **Search Filter** value as needed.
4. Click **Next**.

   Note: PingFederate tries to establish a secure connection to your AD via LDAPS.

   If your AD LDAP server does not support LDAPS, the **Unsecure Connection** screen appears. If you want to continue without a secure connection, click **Next**.

   If the subject of the certificate presented by your AD LDAP server does not match the **Hostname** value, the **Unsecure Connection** screen appears. Click **Previous** to update the **Hostname** field. If you want to continue without a secure connection, click **Next**.

   If the certificate presented by your AD LDAP server is not trusted by PingFederate, the **Certificate Error** screen appears. Import the certificate used by your AD LDAP server to establish a secure connection or select the **I want to complete an unsecure connection** check box to continue without a secure connection, and then click **Next**.

Configure Kerberos authentication

PingFederate is also capable of authenticating users using Active Directory credential tokens (specifically Kerberos service tickets), providing Windows users a seamless single sign-on experience.

Note: If you decide not to enable Kerberos authentication, users will authenticate through the HTML Form Adapter that was automatically created in the previous screen where you connected PingFederate to your Active Directory.

1. Select the **Configure Kerberos Authentication** check box.
2. Enter the realm name, the Kerberos service account and its password.

   Important: If you have not created or configured a service account for Kerberos authentication, see *Configure the Active Directory environment* on page 220 for additional steps. You must have Domain Administrator permissions to make the required changes.
3. Optional: Enter one or more **KDC Hostnames**. If unspecified, PingFederate uses a DNS query to find a list of KDCs.
4. Optional: Click **Test** to verify the connectivity to your KDCs from the administrative console.

   When a connection to any of the KDCs is successful, the message `Test was successful` appears. Otherwise, the test returns error messages near the top of the screen.

   Note that the test stops at the first successful result, so all KDCs are not necessarily verified. Also, connectivity may be subsequently affected in different deployment scenarios, including for engine server nodes running in a clustered environment.
5. Enter one or more **Internal IP Ranges** in CIDR notation to indicate the boundaries of your network.

   End users outside of your network will authenticate through the HTML Form Adapter created in the previous screen.

   Note: End users using mobile clients, such as iPhone and Android mobile phones, will always authenticate through the HTML Form Adapter that was automatically created in the previous screen where you connected PingFederate to your Active Directory.
6. Click **Next**.

Important: You also need to configure the end-user browsers for seamless Kerberos authentication. For more information, see *Configure end-user browsers* on page 512.

Enable provisioning to PingOne

If you have selected to connect PingFederate to PingOne®, the **Provisioning** screen appears and the **Configure Provisioning** check box is selected for your convenience.

This capability gives you the flexibility to provision users from PingOne to SaaS vendors when adding cloud applications later in the PingOne admin portal (see *Add an Application from the Application Catalog* in the PingOne *Employee SSO Administration Guide*).

Users and group provisioning also allows you to configure user access to cloud applications in the PingOne admin portal based on groups and membership information without waiting for end users to sign on (see *Manage Users by Group* in the PingOne *Employee SSO Administration Guide*).

1. Specify the **Group DN** where PingFederate should look for member users (under the **Search Base** previously defined in the Connection screen) to provision to PingOne.

   Note: Groups under the **Search Base** are also provisioned to PingOne automatically.
2. Optional: Select the **Nested** check box to if you want PingFederate to provision users through nested group membership.
3. Click **Next**.

Review your identity provider configuration

On the **Summary** screen, review your configuration.

Note: If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Next** multiple times until you are back to this **Summary** screen.

- Click **Done** to continue.

Create an administrator account

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

Note: If you have selected to connect PingFederate to PingOne®, this screen does not appear because the option to SSO to the PingFederate administrative console from PingOne has been enabled for you.

As needed, you can manage the SSO option in the **Server Configuration** > **Server Settings** > **PingOne Settings** screen after the initial setup.

- Click **Next** to continue.

Review the initial configuration

On the **Confirmation** screen, review your configuration.

Note: If you wish to edit any settings, select the respective screen. If you wish to start over, restart your browser.

On the **Complete** screen, click **Next** to apply the configuration, and then click **Done**.

Install PingFederate as a service

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 the PingFederate service on Windows manually

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.

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 Oracle Java SE Runtime Environment (Server JRE) is installed and the required environment variables are set correctly (see *Install Java* on page 19).
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**.

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

Install the PingFederate service on Linux manually

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.

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 Oracle Java SE Runtime Environment (Server JRE) is installed and the required environment variables are set correctly (see *Install Java* on page 19).
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

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

Uninstall PingFederate from a Windows server

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	1. (Optional) Make a backup copy of the PingFederate installation directory (`<pf_install>`). 2. Use **Control Panel** > **Uninstall a Program** to uninstall PingFederate, which removes the PingFederate service and the installation directory.
Distribution ZIP file	1. Open the **Control Panel** > **Administrative Tools** > **Services** management console. 2. 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. 3. **Optional:** Remove the PingFederate installation directory (`<pf_install>`).

Uninstall PingFederate from a Linux server

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>`).

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.

To access the console:

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.

Once signed on, the administrative console offers multiple menu choices:

- Identity Provider
- Service Provider
- OAuth Server
- Server Configuration

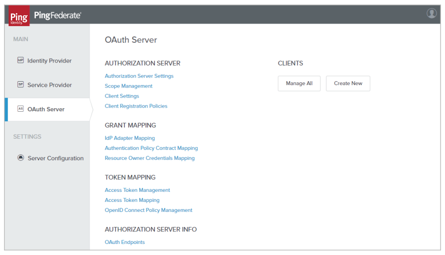
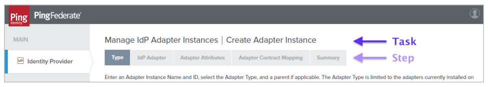


Figure 1: A sample of the OAuth Server menu

Note that the menu choices and menu items varies with the federation roles PingFederate plays (see *Choose roles and protocols* on page 161). Menu items also depend on the permissions assigned to the logged-on administrator (see *Account management* on page 114).

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.



Manage IdP Adapter Instances | Create Adapter Instance

Type | IdP Adapter | Adapter Attributes | Adapter Contract Mapping | Summary

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

Figure 2: 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 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.

This chapter describes:

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 stand-alone 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. Because profiles define the application of the specifications and therefore play a large part in PingFederate, most of the rest of this chapter is devoted to them, starting with *SAML 1.x profiles* on page 35.

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

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.

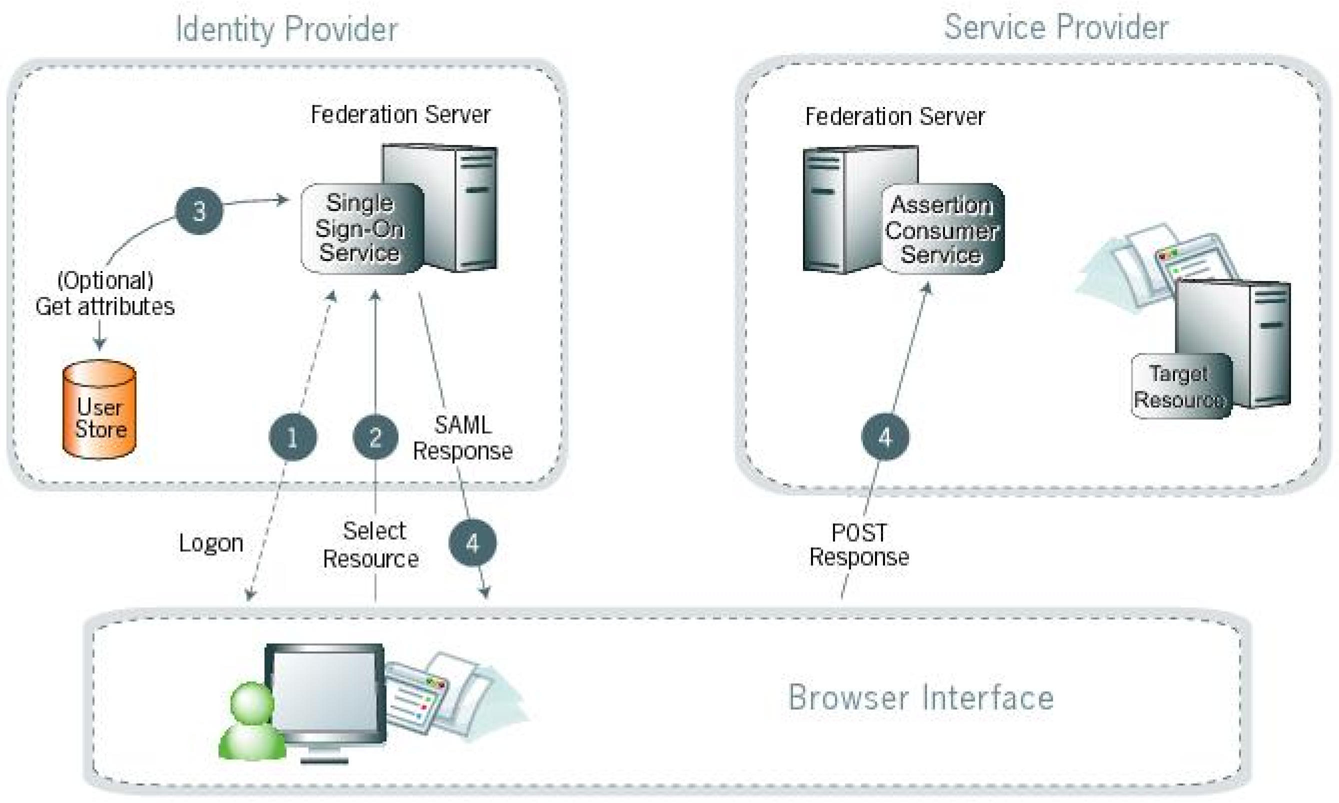


Figure 3: SSO browser/POST profile

Processing 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

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.

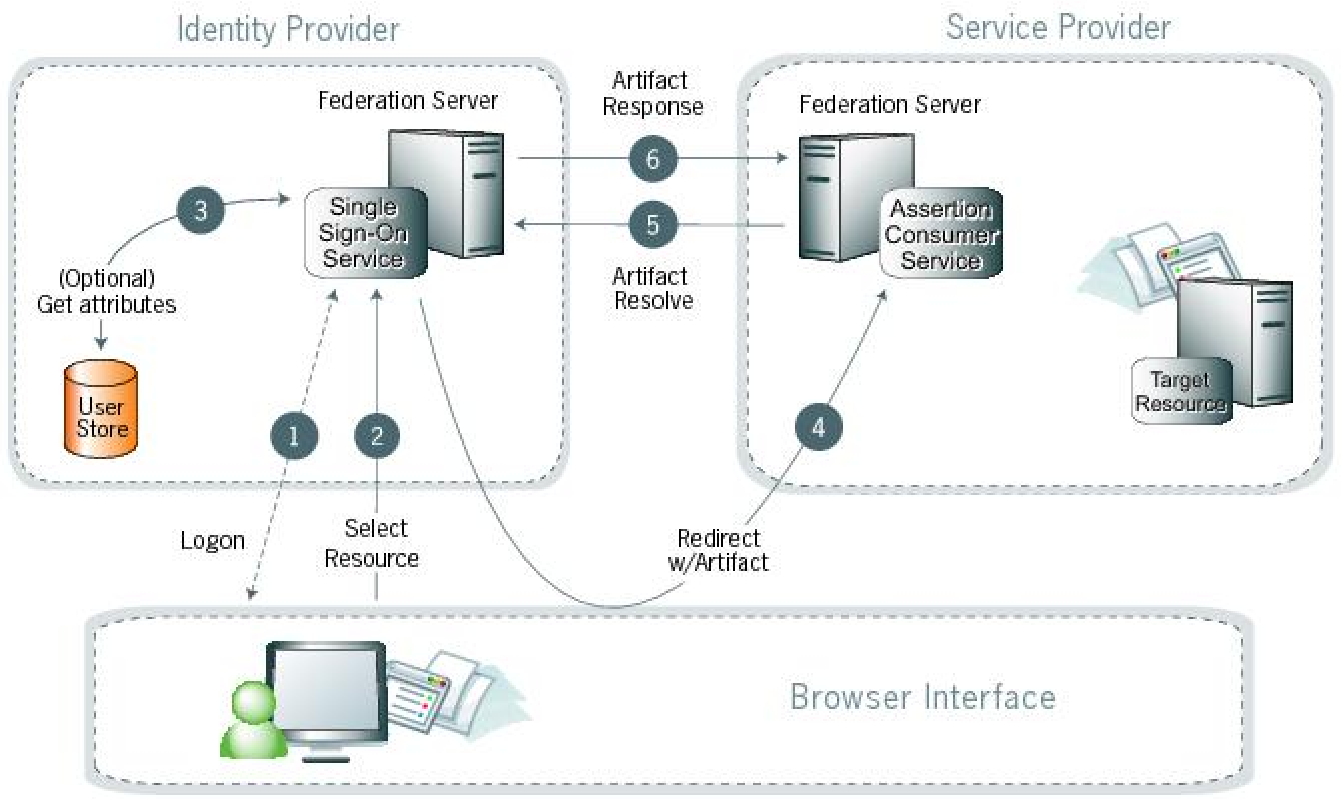


Figure 4: SSO browser/artifact profile

Processing 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

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.

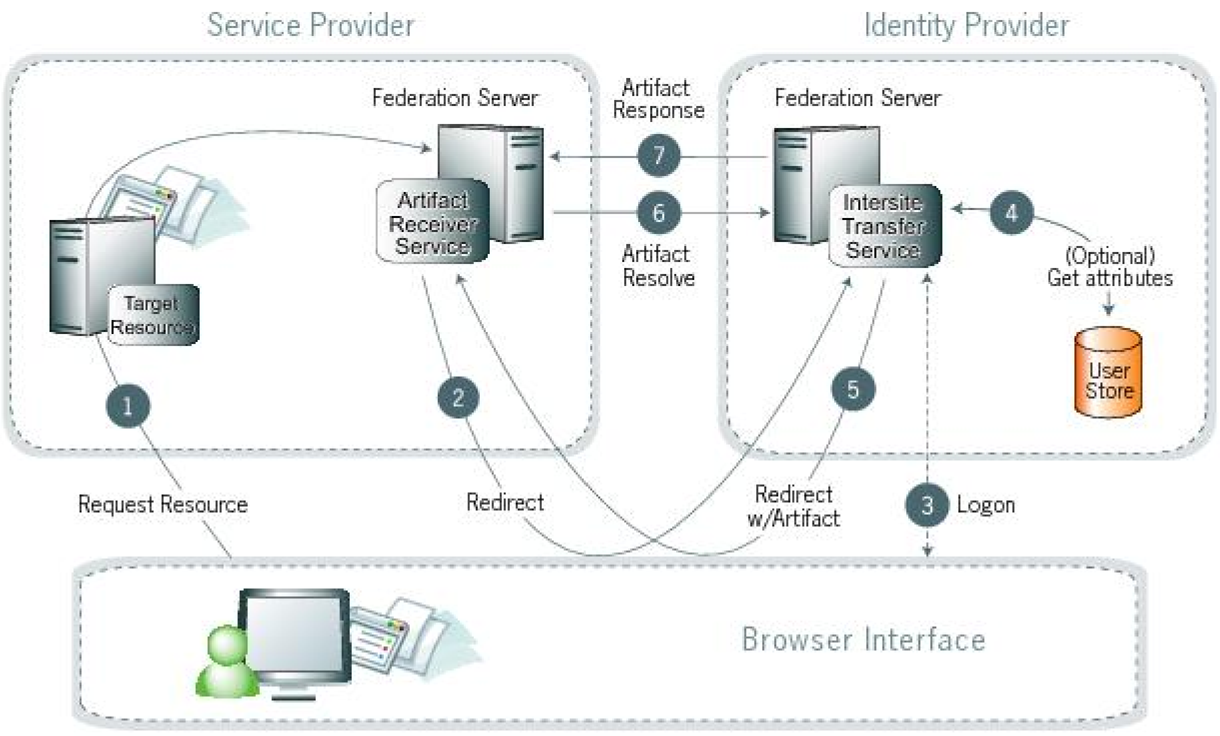


Figure 5: SP-initiated SSO

Processing 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 (e.g., 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 *User attributes* on page 79.)
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* on page 39
- *Single logout* on page 46
- *Attribute Query and XASP* on page 47
- *Standard IdP Discovery* on page 47

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:

- *SP-initiated SSO—POST-POST* on page 39
- *SP-initiated SSO—Redirect-POST* on page 40
- *SP-initiated SSO—Artifact-POST* on page 41
- *SP-initiated SSO—POST-Artifact* on page 42
- *SP-initiated SSO—Redirect-Artifact* on page 43
- *SP-initiated SSO—Artifact-Artifact* on page 44
- *IdP-initiated SSO—POST* on page 45
- *IdP-initiated SSO—Artifact* on page 46

SP-initiated SSO—POST-POST

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.

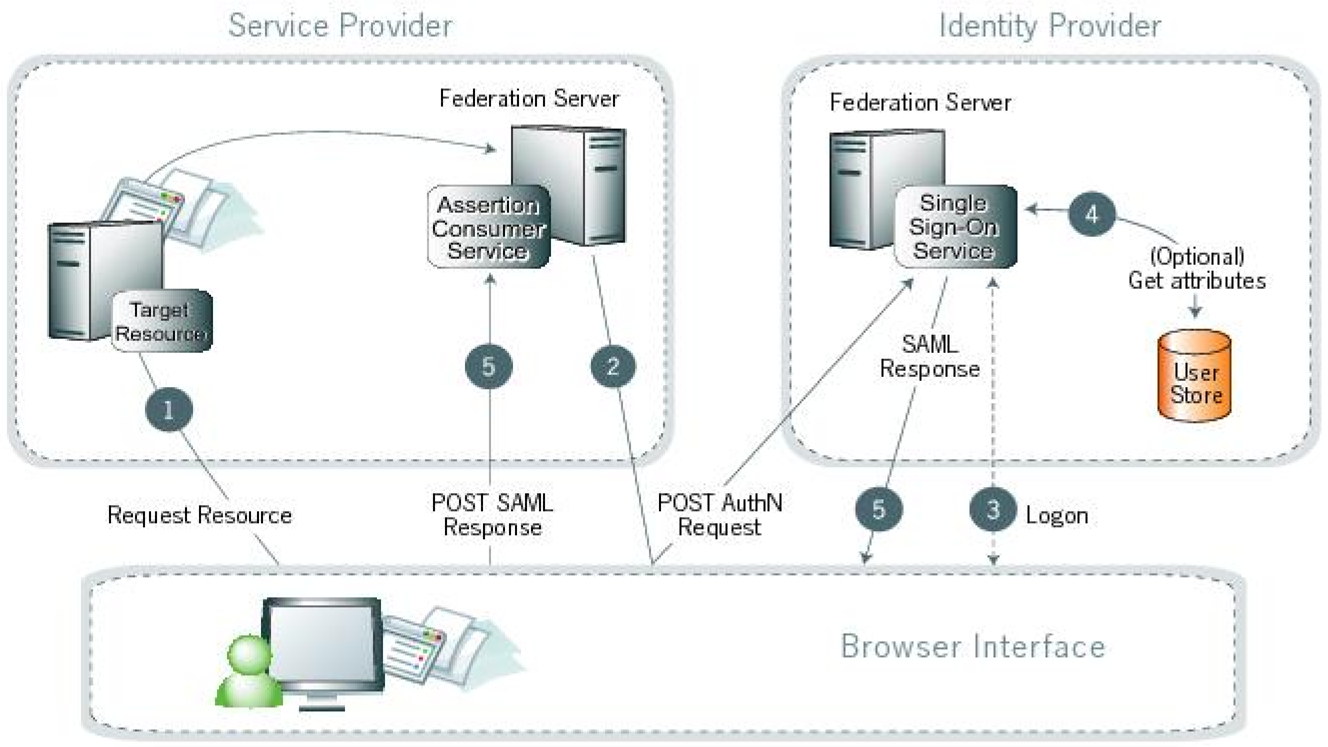


Figure 6: SP-initiated SSO: POST/POST

Processing 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 (e.g., 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 *User attributes* on page 79.)
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

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.

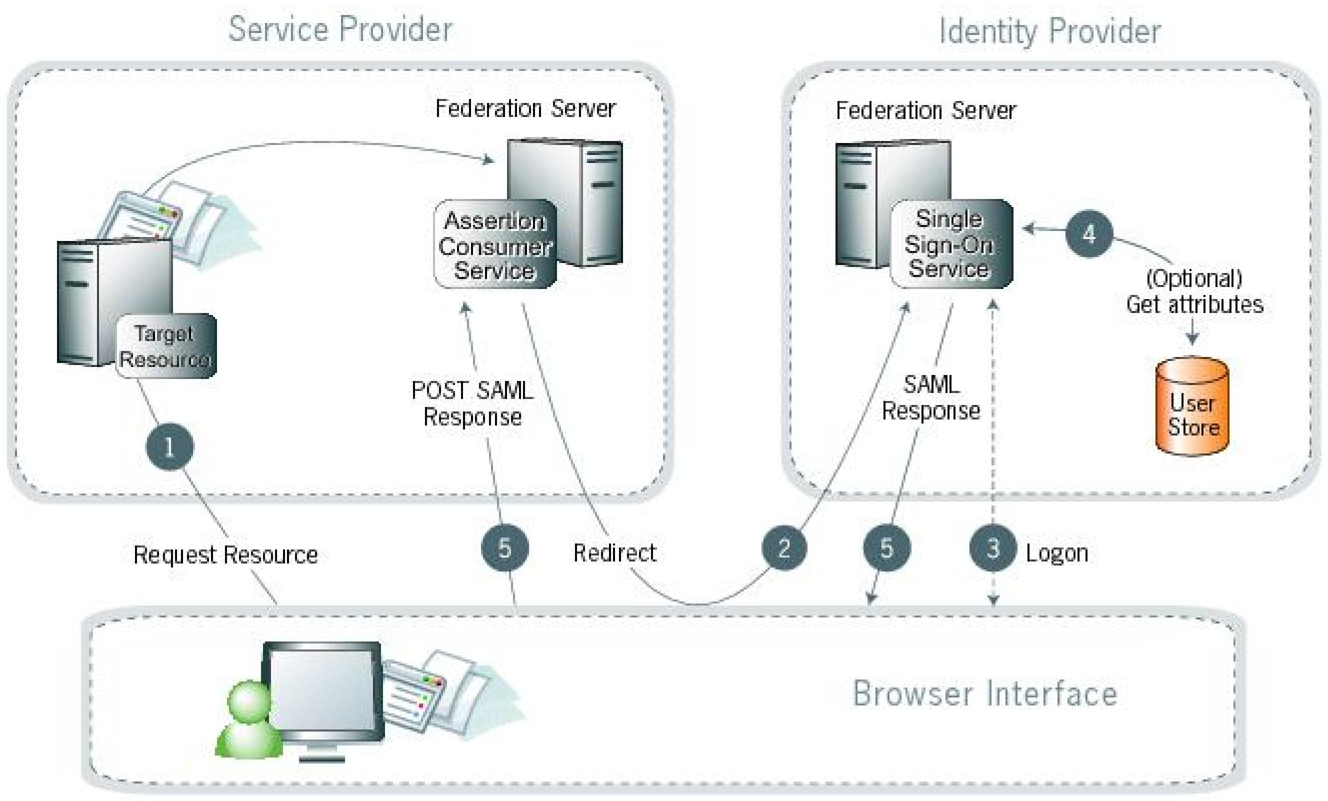


Figure 7: SP-initiated SSO: redirect/POST

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request 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 (e.g., 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 *User attributes* on page 79.)
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

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.

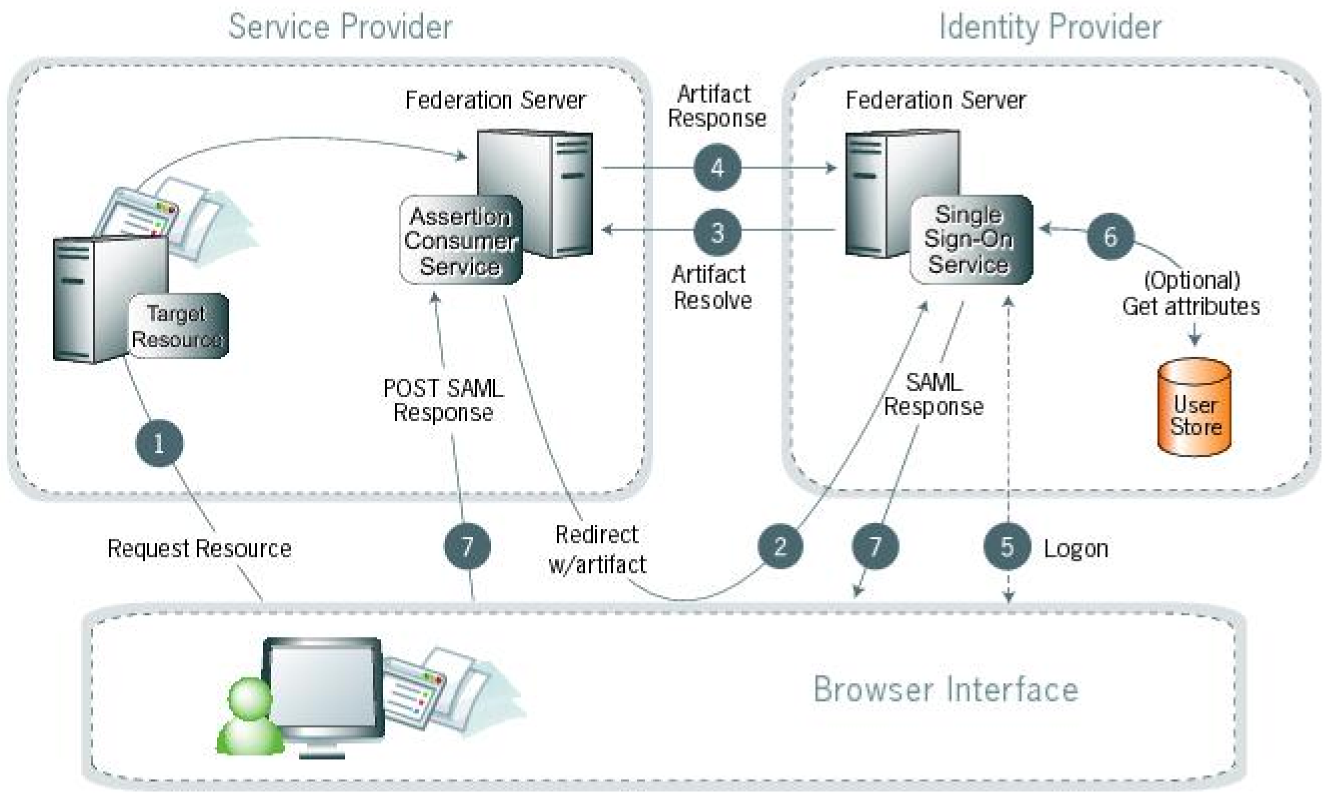


Figure 8: SP-initiated SSO: artifact/POST

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request 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 (e.g., 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 *User attributes* on page 79.)
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

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 .

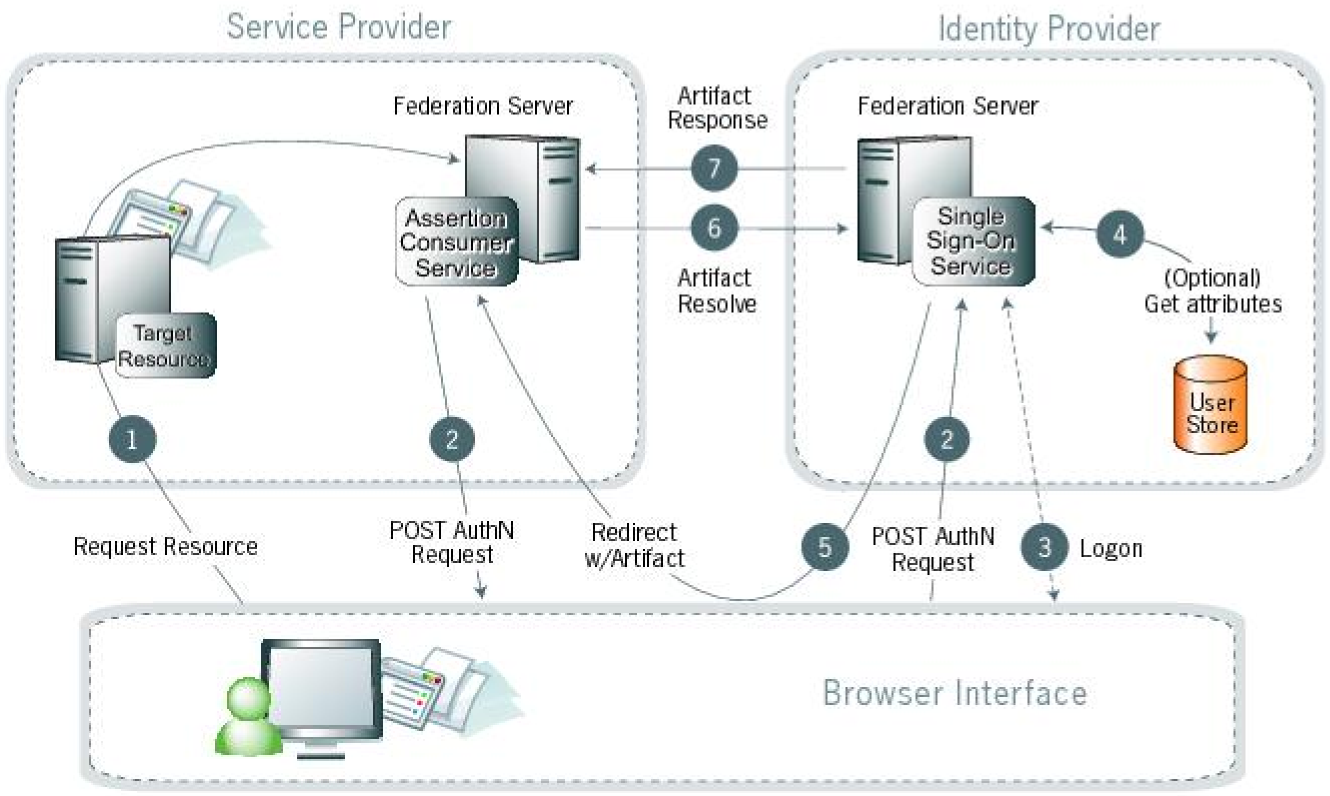


Figure 9: SP-initiated SSO: POST/artifact

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request 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 (e.g., 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 *User attributes* on page 79.)
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

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.

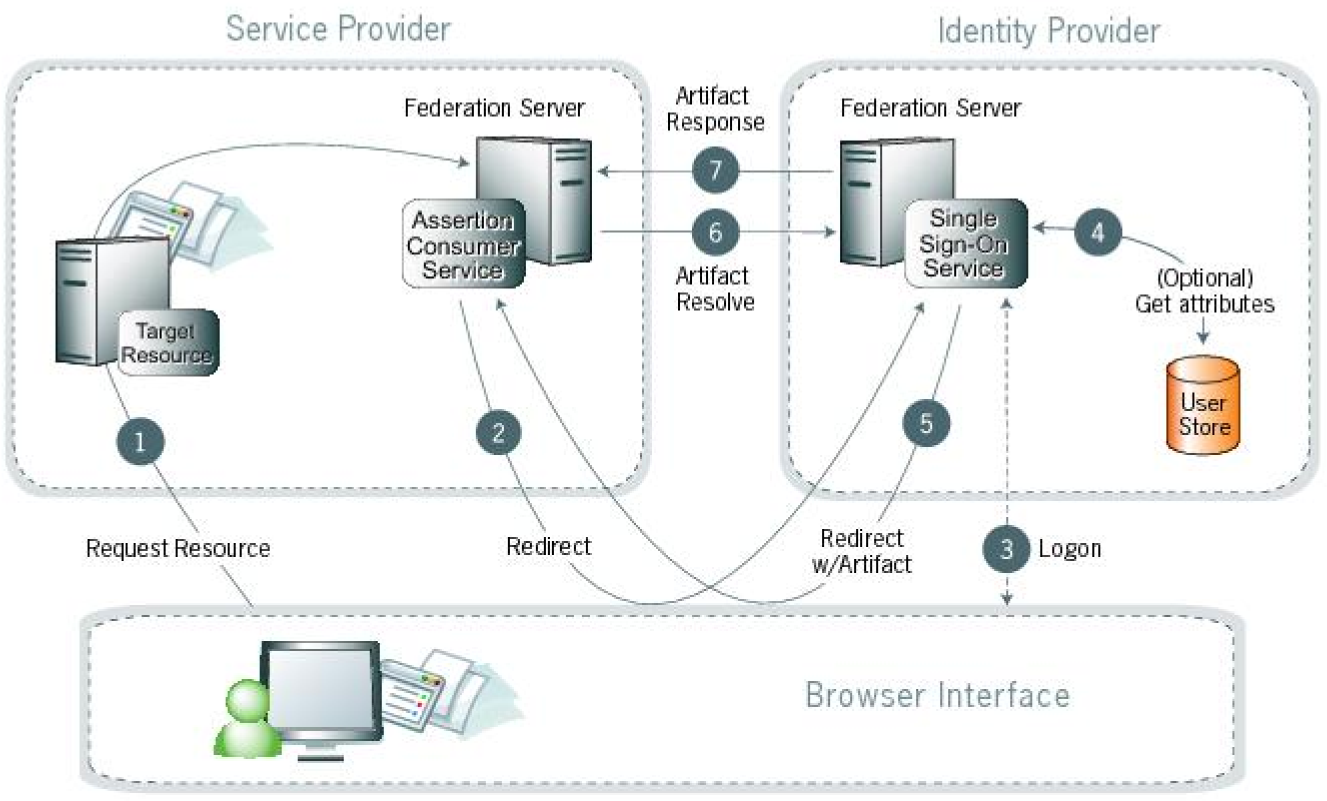


Figure 10: SP-initiated SSO: redirect/artifact

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request 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 (e.g., 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 *User attributes* on page 79.)
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

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.

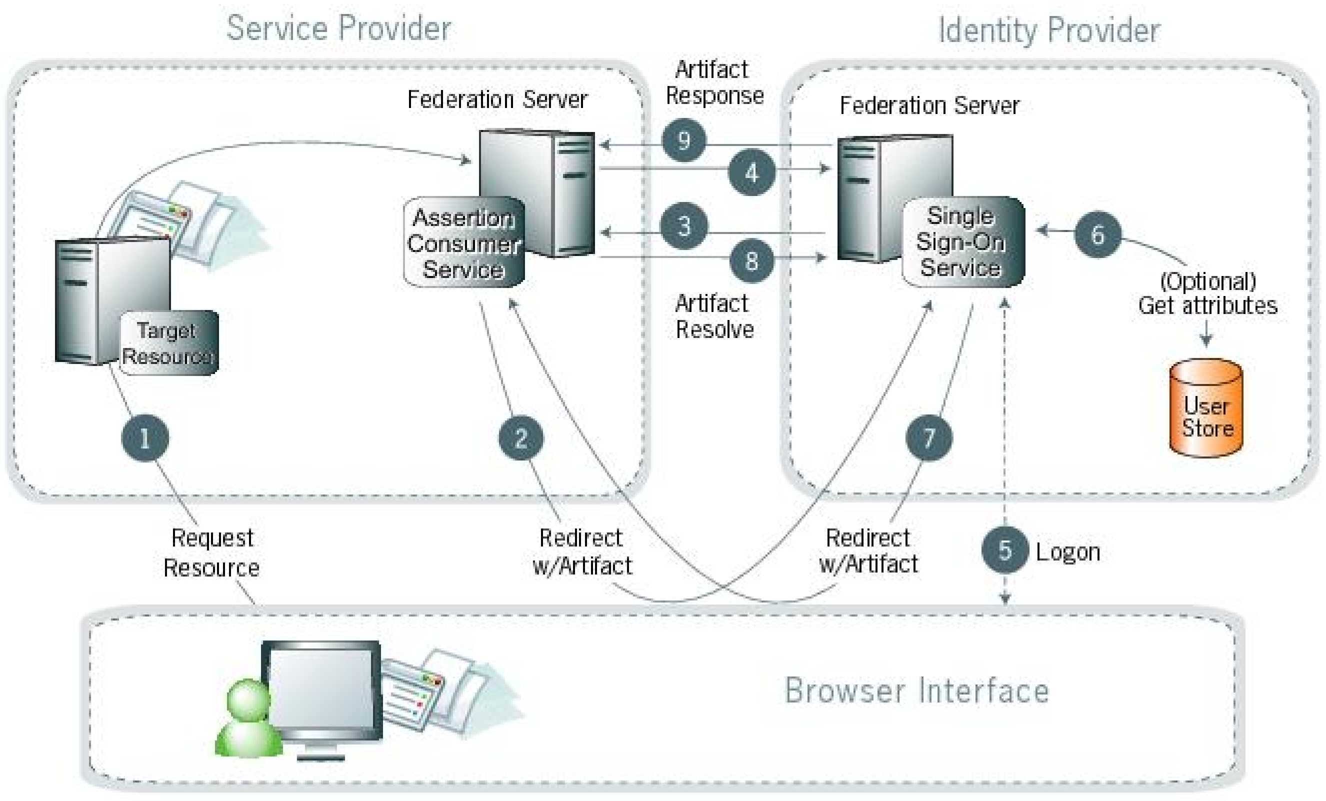


Figure 11: SP-initiated SSO: artifact/artifact

Processing steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request 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 (e.g., 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 *User attributes* on page 79.)
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

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.

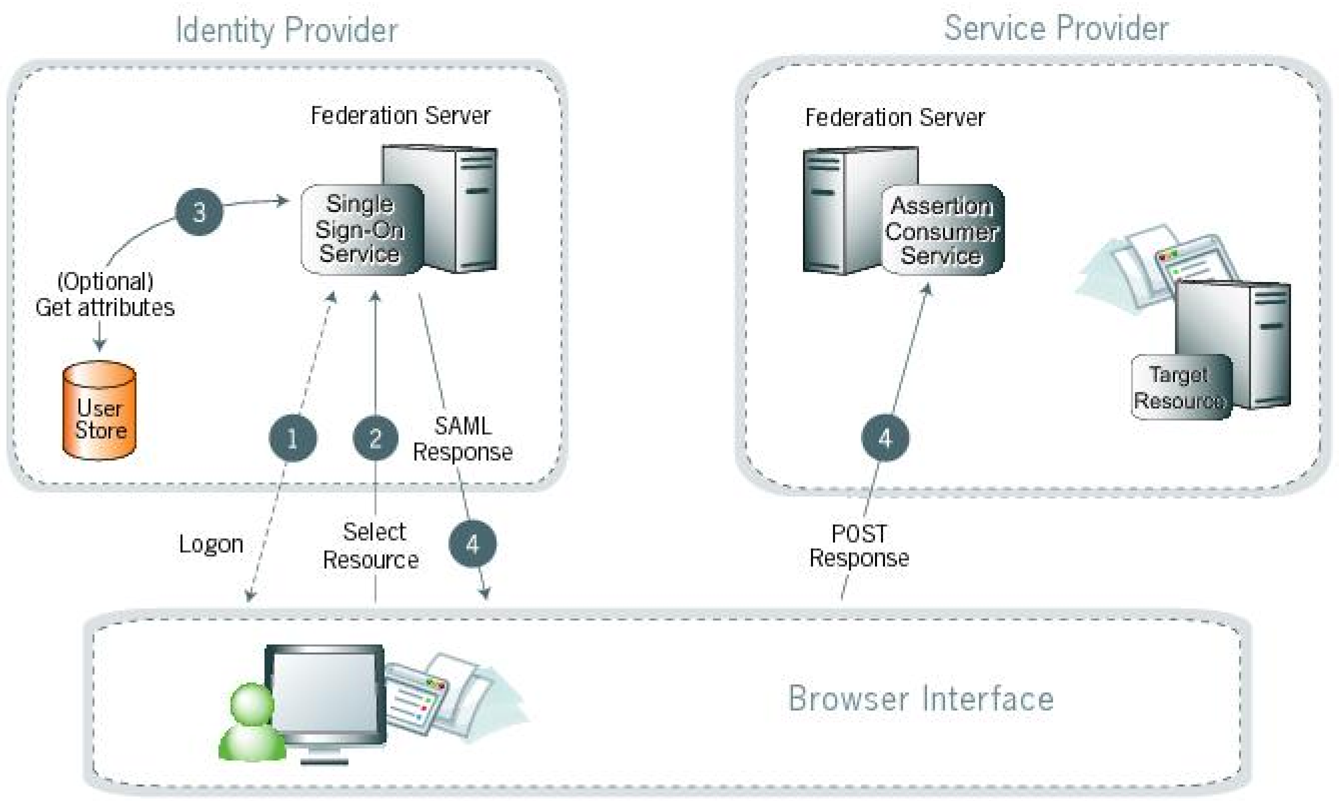


Figure 12: IdP-initiated SSO: POST

Processing 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

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.

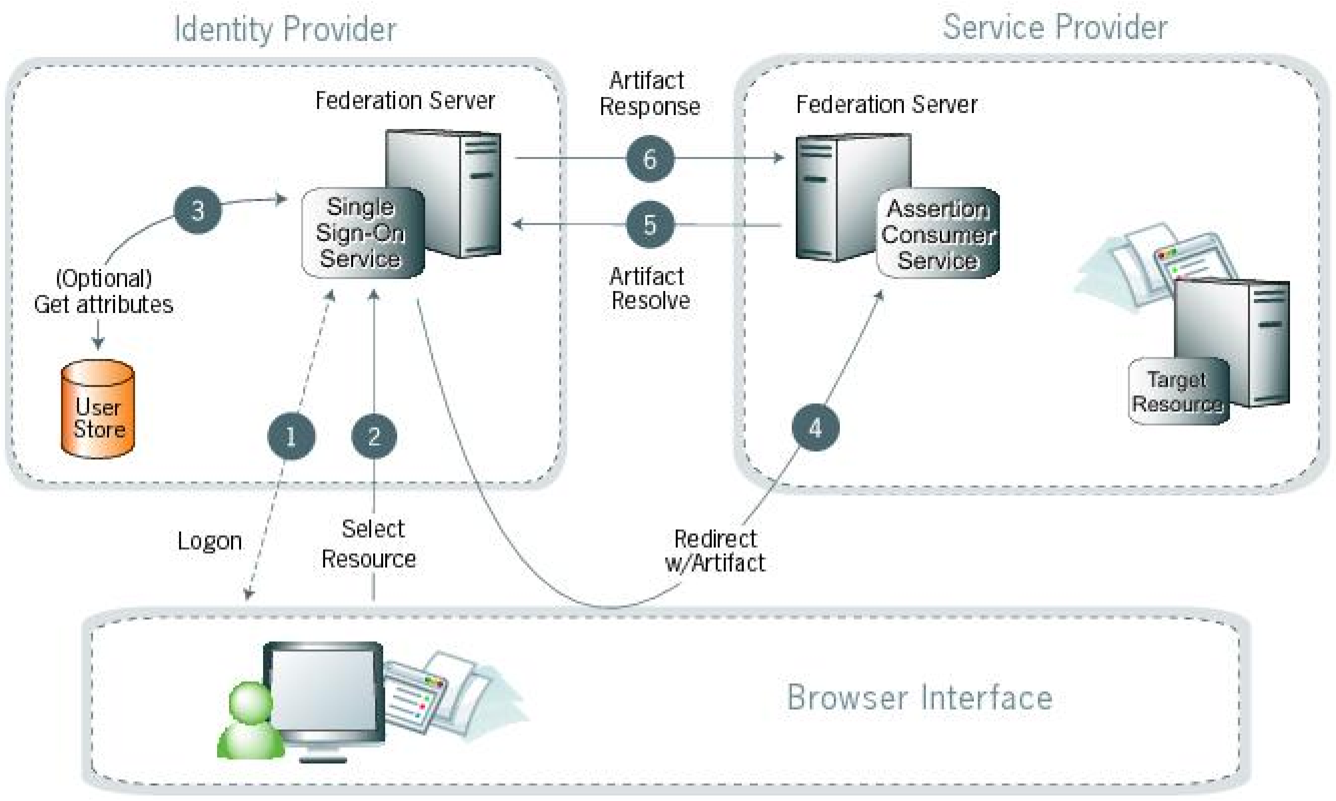


Figure 13: IdP-initiated SSO: artifact

Processing 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 38 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 his or her 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 190.

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
- 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

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).

Passive Requestor profile

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

Figure 19 illustrates message processing for SSO using WS-Federation.

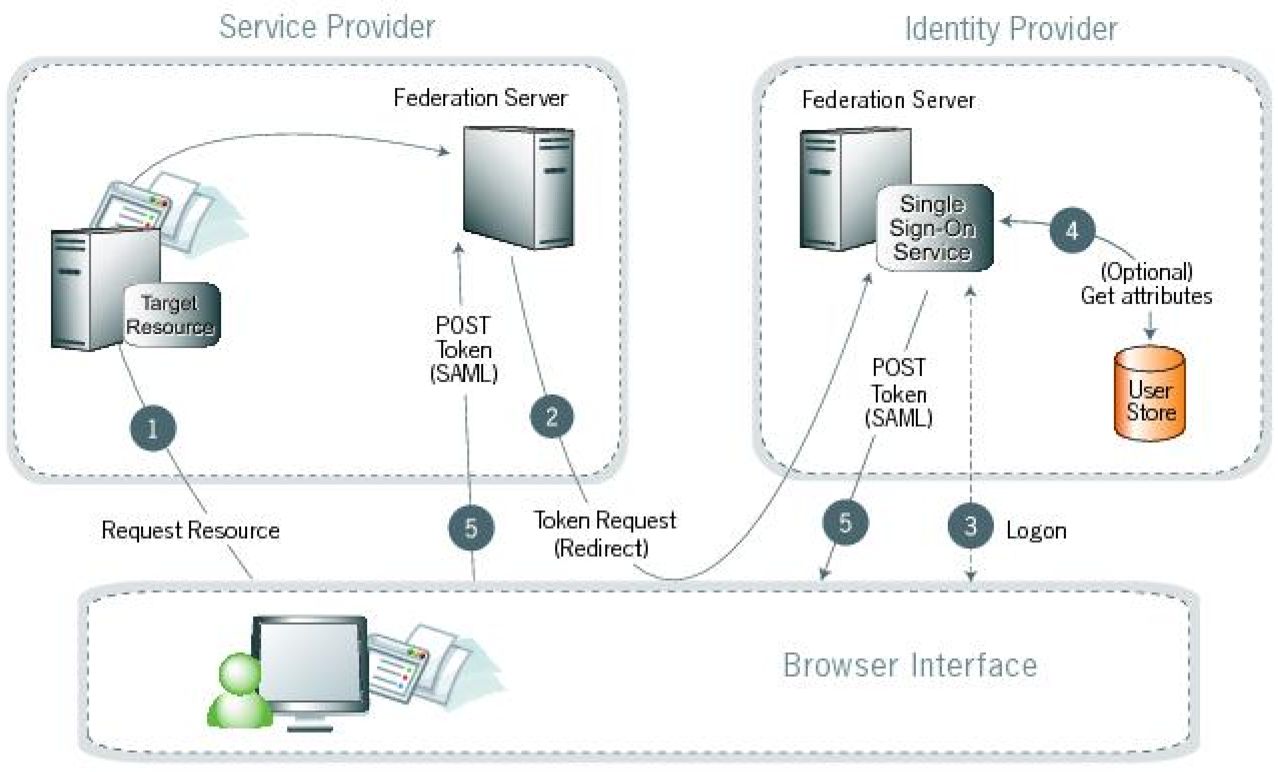


Figure 14: WS-Federation SSO

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request is redirected to the federation server to handle authentication.
2. The SP generates a security token request and redirects the browser to the identity provider's WS-Federation implementation.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (e.g., 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 *User attributes* on page 79.)
5. The federation server creates a response containing a signed SAML assertion (or a JSON Web Token) and returns it to the SP via POST.
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.

Single logout using WS-Federation is handled in much the same way as with SAML (see *Single logout* on page 46); however, HTTP GET/POST is always used as the transport mechanism.

About account linking

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

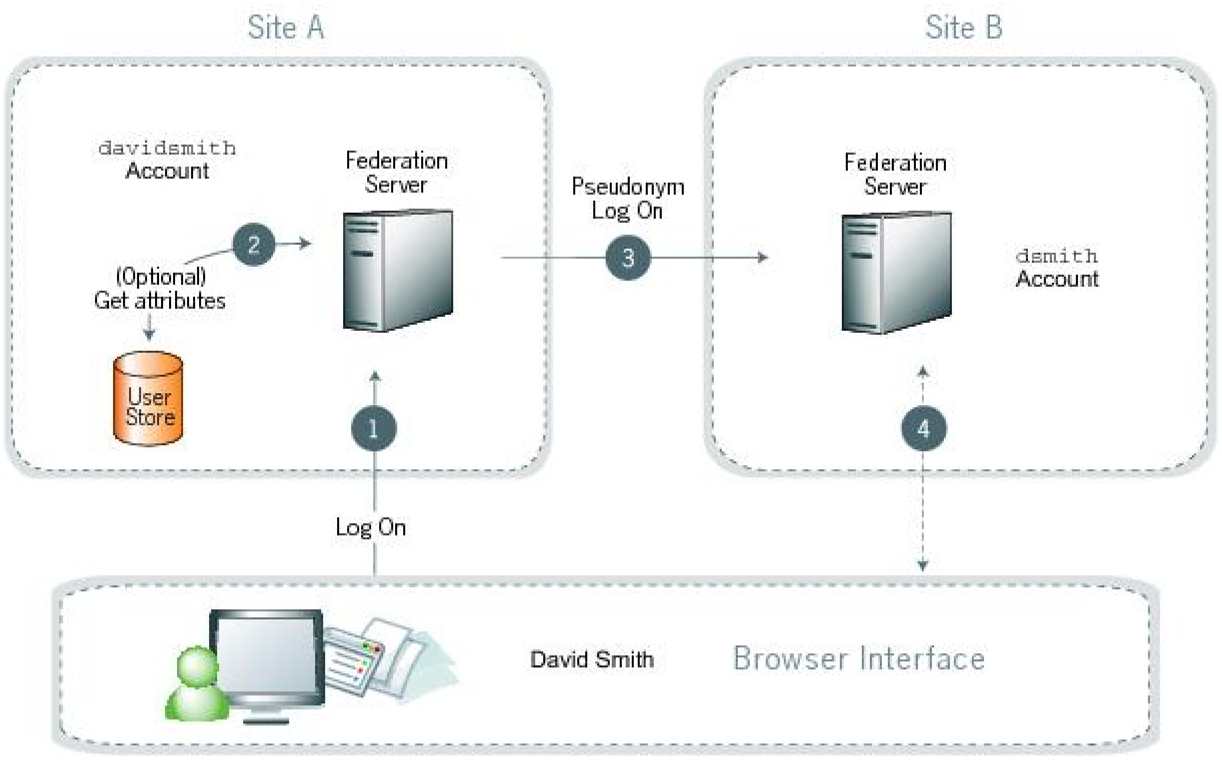


Figure 15: Account linking

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:

Table 1: 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 51).

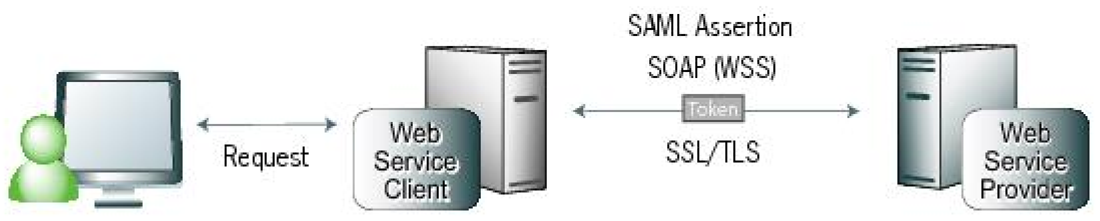


Figure 16: WSS token transfer

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 (see *Token Exchange (Example)*).

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

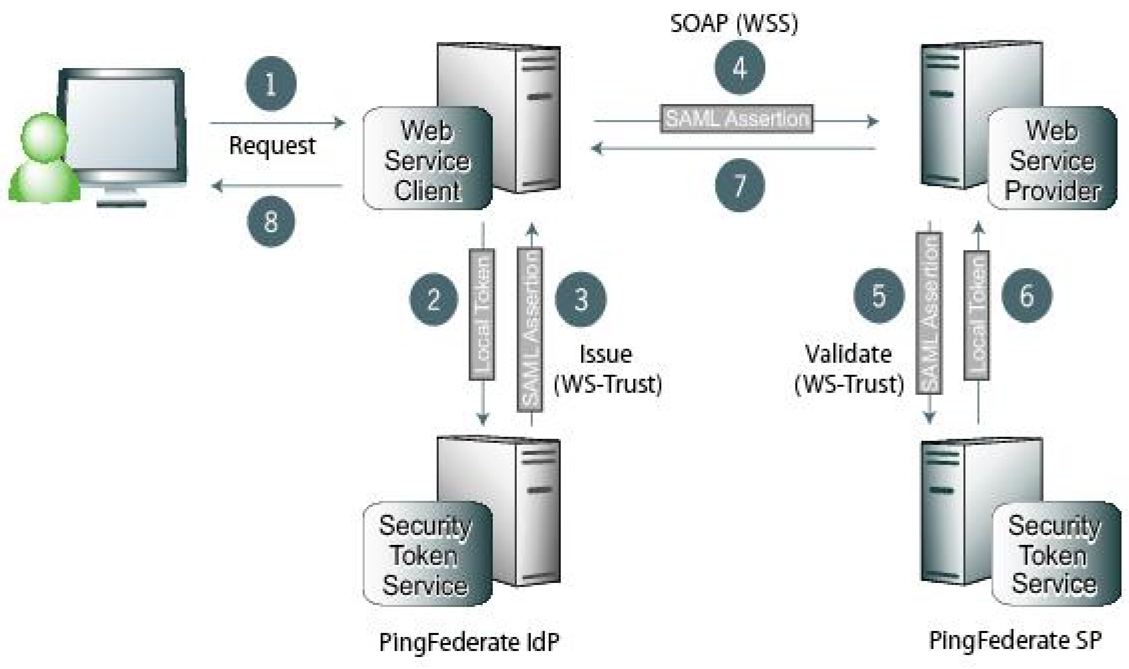
The WS-Trust protocol defines two request types that are particularly useful in securing web services: *Issue* and *Validate*, often associated with the web service client (WSC) and web service provider (WSP), respectively.

- 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.

Figure 17: 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.

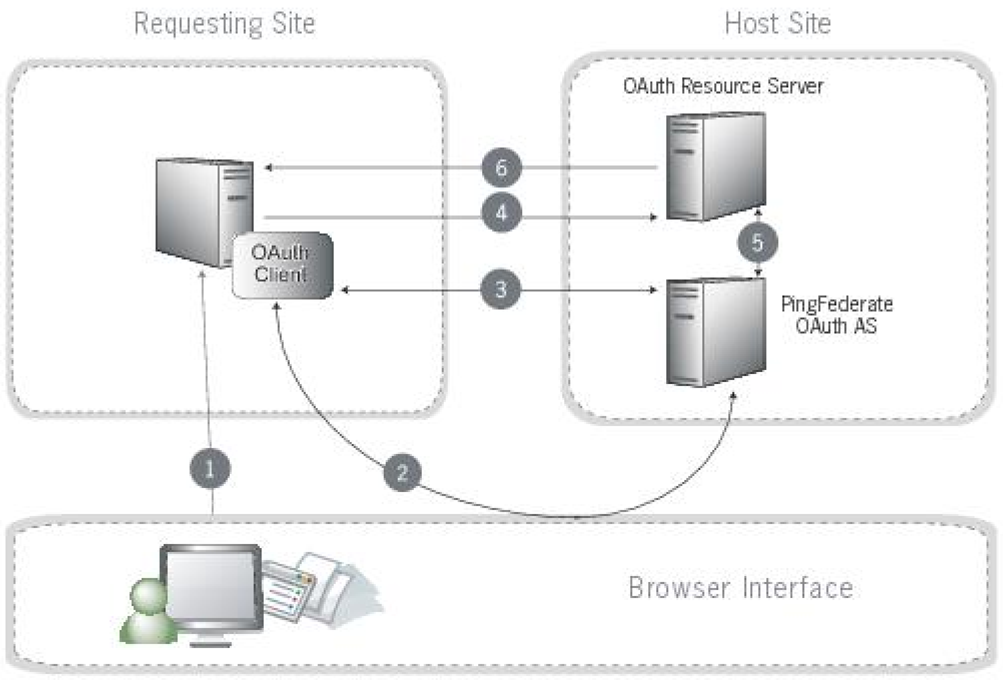


Figure 18: Web redirect flow

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 provides a **Request for Approval** page for the user to authorize the sharing of information.

   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. (Not shown.) The requesting site displays data from the API call to the user.

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.

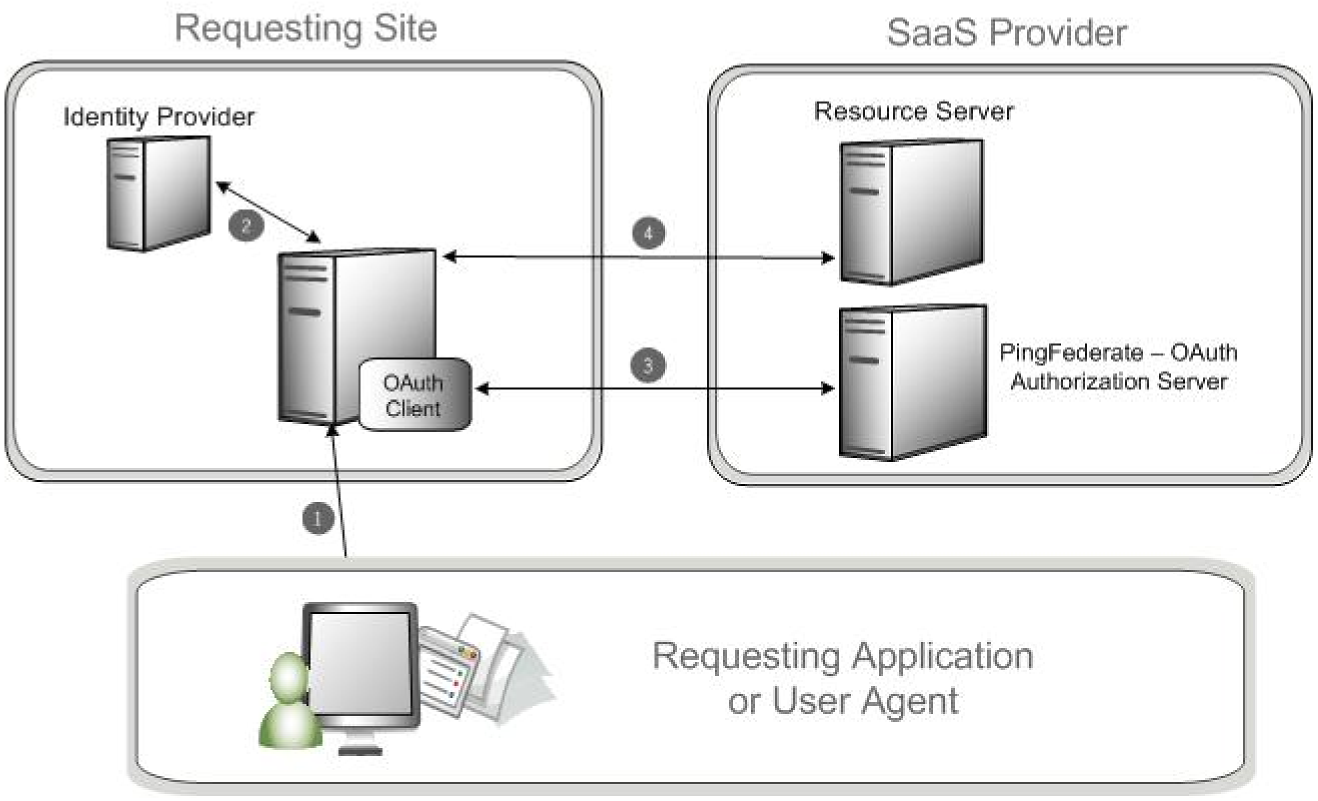

Requesting Site
SaaS Provider
Identity Provider
Resource Server
2
4
OAuth Client
3
PingFederate – OAuth Authorization Server
1
Requesting Application or User Agent

Figure 19: 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 custom 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).

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:

- Gemalto SafeNet Luna SA HSM
- Thales 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.

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.

Install and configure Gemalto SafeNet Luna SA client and PingFederate

1. Install and configure your Gemalto SafeNet Luna SA 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.
2. 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 or UNIX	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 contact SafeNet documentation for more information.

3. Edit 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
```

4. 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.

5. Edit the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory and update the `<!-- Crypto provider -->` section.
   a) For the JCEManager service endpoint, change the value of the construct class to as follows:

   ```
   <construct class="com.pingidentity.crypto.LunaJCEManager5"/>
   ```

   b) For the CertificateService service endpoint, change the value of the construct class as follows:

   ```
   <construct class="com.pingidentity.crypto.LunaCertificateServiceImpl5"/>
   ```

6. Edit 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 208.

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

   Enter the NTL password when prompted (see *step 1*).

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

   Note: The Gemalto SafeNet Luna SA 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
   ```

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

This completes the steps required to configure PingFederate for use with the Gemalto SafeNet Luna SA. 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 SA operational notes

Some restrictions apply to PingFederate operations when using an HSM:

- 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.

Install and configure Thales nShield Connect client and PingFederate

1. Install and configure your Thales nShield Connect HSM client software.

   As part of the installation, install the optional Java Support (including KeySafe) and nCipherKM JCA/JCE provider classes components.
2. After your installation, refer to the Thales nShield documentation 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.
3. If you have not already done so, download and install the (JCE) Unlimited Strength Jurisdiction Policy Files 8 (`jce_policy-8.zip`).

   Follow instructions in the readme to install the Policy Files.
4. To enable the Java interface, copy the `NFAST_HOME/java/classes/nCipherKM.jar` file to the `JAVA_HOME/jre/lib/ext` directory.

   Thales 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 Thales documentation for more information.
5. Edit 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. Edit the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory and update the `<!-- Crypto provider -->` section.
   a) For the JCEManager service endpoint, change the value of the construct class to as follows:

      `<construct class=`**`"com.pingidentity.crypto.NcipherJCEManager"/>`**

   b) For the CertificateService service endpoint, change the value of the construct class as follows:

      `<construct class=`**`"com.pingidentity.crypto.NcipherCertificateServiceImpl"/>`**

8. Edit 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 208.

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

   Enter the Operator Card Set password when prompted (see *step 2*).

   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 Thales documentation for instructions on restarting the hardserver.)
    e) Log on to the PingFederate administrative console, go to the **Server Configuration** > **Cluster Management** screen and then click **Replicate Configuration** to push the configuration changes, which includes the nShield data, to the engine nodes.

This completes the steps required to configure PingFederate for use with the Thales nShield Connect. 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 operational notes

Some restrictions apply to PingFederate operations when using an HSM:

- When PingFederate is integrated with Thales nShield Connect on a platform with Oracle Java SE Runtime Environment (Server JRE) 8 update 102, runtime errors may occur when handling certificates with a signing algorithm of RSA SHA256, SHA384, or SHA512. Upgrading to Oracle Server JRE 8 update 112 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.

The Administrator's manual consists of:

- *Key concepts* on page 62 — A discussion of central concepts needed to understand Internet SSO, the WS-Trust Security Token Service (STS), and PingFederate deployment and administration.
- *System administration* on page 96 — Information about maintaining the PingFederate server and deployment, using log files, managing users, and handling other administrative functions.
- *System settings* on page 154 — How to configure your local PingFederate server settings.
- *Security management* on page 192— Information about importing, exporting, and maintaining certificates and keys in PingFederate.
- *Authentication policies* on page 222— Configure Authentication Policies, Selectors, and Policy Contracts to fulfill various authentication requirements, to deploy PingFederate as a Federation Hub, or both.
- *OAuth configuration* on page 264 — How to configure PingFederate to act as an OAuth authorization server.
- *Identity provider SSO configuration* on page 322— How to configure PingFederate to act as an Identity Provider (IdP) and establish connections to Service Providers.
- *Customer IAM configuration* on page 379— How to configure customer identity and access management (customer IAM) use cases in PingFederate.
- *Service provider SSO configuration* on page 400 — How to configure PingFederate to act as a Service Provider (SP) and establish connections to Identity Providers.
- *WS-Trust STS configuration* on page 470 — How to configure PingFederate to act as a Security Token Service for web service clients and Providers in either an IdP or SP environment.
- *IdP-to-SP bridging* on page 490 — Configure advanced federation use cases, such as Adapter-to-Adapter Mappings (for Browser SSO) and Token Exchange Mappings (for STS).
- *Kerberos Adapter* on page 510 — How to configure PingFederate to use the packaged Kerberos Adapter.
- *HTML Form Adapter* on page 498 — How to configure PingFederate to use the packaged HTML Form Adapter.
- *HTTP Basic Adapter* on page 508 — How to configure PingFederate to use the packaged HTTP Basic Adapter.
- *OpenToken Adapter* on page 513 — How to configure PingFederate to use the packaged OpenToken Adapter for interfacing with your web applications.
- *Composite Adapter* on page 519 — How to configure PingFederate to use the packaged Composite Adapter.
- *Application endpoints* on page 527 — Detailed information about using PingFederate connection endpoints for web single sign-on and single logout.
- *OAuth 2.0 endpoints* on page 545 — Detailed information about using PingFederate connection endpoints for the OAuth.
- *Web service interfaces* on page 567 — Information developers can use to automate connection-configuration management and runtime SSO partner discovery.
- *Attribute mapping expressions* on page 593 — How to enable and use expressions in conjunction with mapping attributes.
- *Customize assertions and authentication requests* on page 598 — Information about assertion and authentication request customizations.
- *Fulfillment by data store queries* on page 603 — How to configure data store queries to fulfill a contract or to pull attributes to be verified in the token authorization process
- *Troubleshooting* on page 610 — Solutions for difficulties that may be encountered.
- *Glossary* on page 621 — Definitions of terms used in the manual and in identity federation parlance.
- *List of acronyms* on page 630 — A list of acronyms.

Key concepts

This chapter 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 9.

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 33).
- 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 62). 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 65).
- Provisioning – Provides automated cross-domain inbound and outbound user management (see *User provisioning* on page 84).

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 50.)

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 63).

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 50).

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 72).

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 494).

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 77).

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 66). (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 *STS namespaces* on page 80).

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.

Related concepts
Access token management on page 300

Grant types

To obtain an access token, a client interacts with an OAuth AS, sending a request for an access token that includes an access grant. An access grant is also used when an RS requests validation of an access token from the OAuth AS.

Primary grant types

OAuth defines several different access grant types—each reflecting different authorization mechanisms.

Authorization code

An authorization code is returned to the client through a browser redirect after the resource owner gives consent to the OAuth 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 credentials

The client collects the resource owner's password and exchanges it at the OAuth 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

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 OAuth 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

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 OAuth AS) such as client applications implemented in a browser using a scripting language like JavaScript.

Client credentials

The client presents its own credentials to the OAuth 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:

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 the OAuth AS. The grant type allows an RS to check with the OAuth AS on the validity of a bearer access token received from a client making a protected-resources call.

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 OAuth AS. 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.

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 OAuth AS. Similar to SAML 2.0 Bearer, 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.

Grant storage and management

PingFederate uses its internal HSQLDB database by default to maintain persistent grants for the OAuth AS. Administrators can also configure PingFederate to store persistent grants externally, on a database server, an LDAP server, or a custom storage medium through the PingFederate SDK.

Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked. Expired grants are automatically removed once a day. As needed, administrators can fine-tune the frequency and the number of expired grants to be removed. For revocation, PingFederate provides two endpoints. 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.

Scopes

Where OAuth provides a mechanism to constrain the privileges associated with an access token, scopes provide a way to more specifically define the privileges requested and granted. Generally, a client specifies the scopes desired when asking for authorization. The issued access token is associated with the approved scopes.

Scopes are configured globally using the **OAuth Server** > **Scope Management** configuration wizard but may be customized on a client-by-client basis.

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. Client records are part of the configuration archive.

Alternatively, administrators can configure PingFederate to store client records externally, on a database server, an LDAP server, or a custom storage medium through the PingFederate SDK. 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.

Persistent vs. transient grants

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

Transient grants are valid only for the lifetime of the access token itself. The **Client Credential** access grants, for example, are considered transient.

Persistent grants can result from the following OAuth use cases:

- OAuth clients using the **Refresh Token** grant type in conjunction with either the **Authorization Code** or **Resource Owner Credentials** 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.
- OAuth clients using the **Implicit** grant type *and* the **Reuse Existing Persistent Access Grants for Grant Types** check box is selected in the **OAuth Server** > **Authorization Server Settings** screen.

  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.

Support for persistent grants requires PingFederate to use a database server or an LDAP directory server for long-term storage. The data structure contains a USER_KEY attribute, which can be populated according to information mapped using attributes obtained during a user's initial authentication verification within PingFederate.

PingFederate automatically removes expired grants and the associated attributes once a day. As needed, you can fine-tune the frequency and the size of the cleanup batch.

Important: PingFederate uses a pre-installed HSQLDB database as its grant data store for initial setup and testing. We however strongly recommend that you choose your own, secured database for production deployments.

PingFederate also supports other storage solutions through the PingFederate SDK. For more information, see the Javadoc for the `AccessGrantManager` interface.

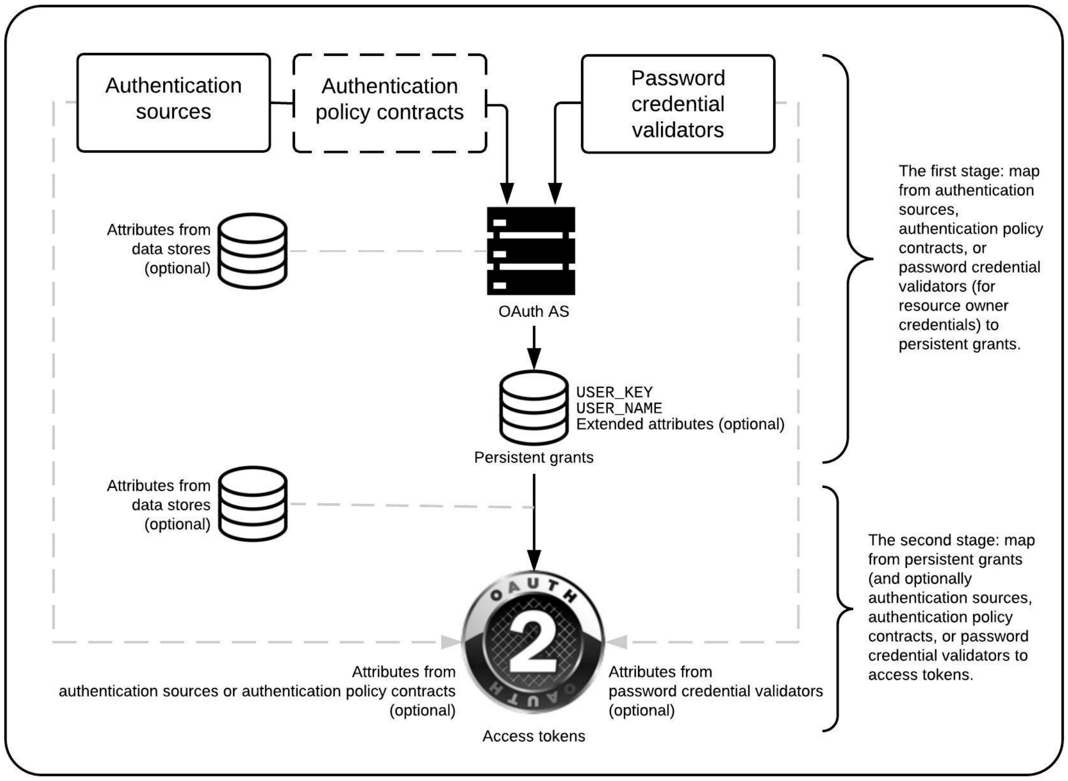
Tip: The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.

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.

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.

Related tasks
Grant mapping on page 285
Token mapping on page 299

OAuth user-facing screens

The PingFederate OAuth AS provides two screens presented to end users (the resource owners) during OAuth transactions, one requesting approval of the scope of protected resources requested and one providing a means of revoking persistent access grants. 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/JWKS`
- `/.well-known/openid-configuration`

As needed, administrators can add or remove allowed origins using the administrator console. 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 stand-alone 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 656 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 commercial 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, for IdP and federation hub deployments PingFederate provides plug-in *authentication selectors*, which enable dynamic selection of authentication sources (IdP adapter instances, or IdP connections for federation hub use cases) based on administrator-specified criteria (see *Bundled authentication selectors* on page 73).

PingFederate also includes a robust software development kit (SDK), which software developers can use to write their own adapters for specific systems. Adapters can be written to retrieve attributes from custom data stores, connect to application- or IdM-specific user authentication systems, or provide complex attribute transformations or processing.

Bundled adapters

PingFederate comes bundled with a set of adapters.

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 510).

HTML Form Adapter and HTTP Basic Adapter

Used in conjunction with Password Credential Validators (see *Manage Password Credential Validator instances* on page 211). These adapters provide integration, for example, with LDAP user-data stores such as Active Directory or direct user logon via credentials maintained by PingFederate (see *HTML Form Adapter* on page 498 and *HTTP Basic Adapter* on page 508).

OpenToken Adapter

Provides a generic interface for integrating with various applications, including Java- and .NET-based applications (see *OpenToken Adapter* on page 513).

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 519).

Bundled authentication selectors

For IdP and federation hub deployments, PingFederate also includes global (cross-connection) authentication selectors. Along with the Composite Adapter (see the previous section) 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 may be used to select subsequent selectors or authentication sources, which allows handling of complex hierarchical access-policy decisions (see *Authentication policies* on page 222).

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 *Configure the CIDR Authentication Selector* on page 223).

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 *Configure the Cluster Node Authentication Selector* on page 224).

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 *Configure the Connection Set Authentication Selector* on page 224).

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 *Configure the HTTP Header Authentication Selector* on page 224).

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 *Configure the HTTP Request Parameter Authentication Selector* on page 225).

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 *Configure the OAuth Scope Authentication Selector* on page 227).

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 (see *Browser-based SSO* on page 35). Configured authetication sources are mapped either to SAML-specified contexts or any ad-hoc context agreed upon between the IdP and SP partners (see *Configure the Requested AuthN Context Authentication Selector* on page 227).

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.

Commercial adapters and selectors

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 663.

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 198).

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 207.

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 a specific certificate 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.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 import control restrictions, the standard Java Runtime Environment (JRE) distribution supports strong but not unlimited encryption. For this reason, the strongest cipher suites in the same configuration files are also commented out. To use the strongest encryption, when permissible, remove the comments from the AES 256 cipher suites and download and install the appropriate version of "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" from the *Oracle download website* (www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html).

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 *Manage cipher suites* on page 149).

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 196).

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 198).

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 63)
- Access Token Management instances (see *Access token management* on page 300)
- Password Credential Validators (see *Manage Password Credential Validator instances* on page 211)
- Identity Store Provisioners (see *Configure Identity Store Provisioners* on page 403)

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 more information about SAML, see *Supported standards* on page 33.)

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

- *Account linking* on page 78
- *Account mapping* on page 78

For WS-Trust STS, account mapping is used.

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 *Define an account-linking data store* on page 177.

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 535). A user who has defederated may later elect to reassociate 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 *Define SP affiliations* on page 376). 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 83).

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 593)

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 401 or *Manage token generators* on page 483). 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 80 below or *STS token contracts* on page 81).

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 340).

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 477). (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 *Windows Identity Foundation clients* on page 64). 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 477).

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 72 and *Manage IdP adapters* on page 323).

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 401.)

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 419).

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 63 and *Manage token processors* on page 472).

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 483). 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 486).

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

PingFederate can be configured to use data stores to supply attributes for various contracts (for example, adapter contracts, browser-based SSO token contract, or STS token contracts) or conditions to be verified in the token authorization process. PingFederate supports a wide varierty of directory servers and database servers for user-attribute lookup. Alternatively, you can create your own driver for non-JDBC or non-LDAP data stores, such as flat files or SOAP-connected databases, using the PingFederate Custom Source SDK, and then map from custom fields to fulfill your use cases.

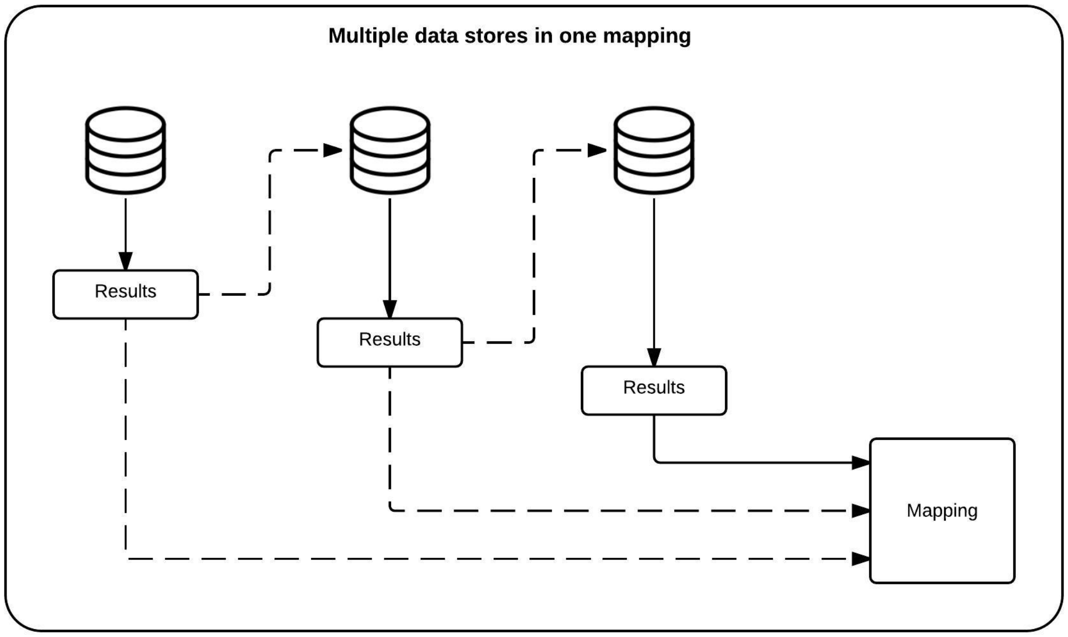
Data stores can be used across multiple configurations.

Multiple data source attribute mapping

When querying data stores for attributes to help fulfill a contract on the IdP side, PingFederate can be configured to use more than one attribute source.

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*.

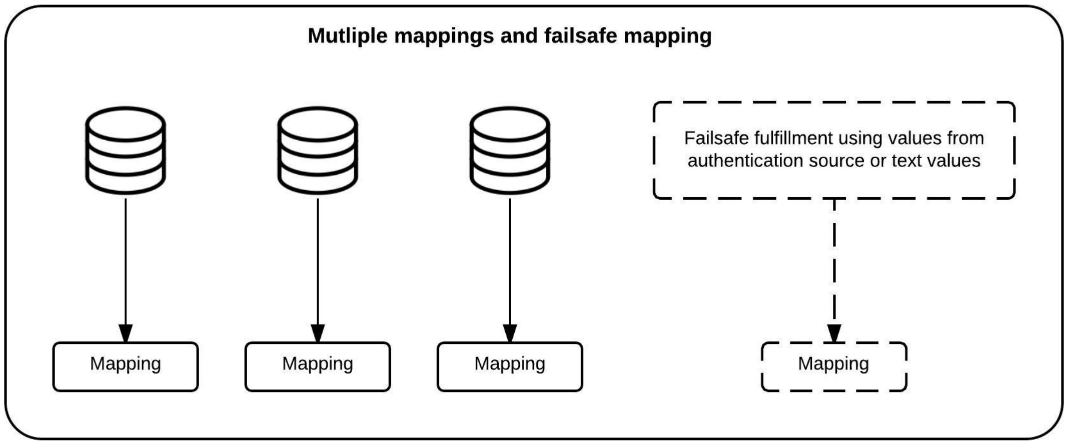
If a required attribute (such as SAML_SUBJECT in a SAML contact or subject in an authentication policy contract) cannot be fulfilled, the request fails.

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:

- OpenID Connect policies (the user-attributes mapping process)
- Resource-owner credential mappings
- IdP adapter mappings
- Access token mappings

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

Attribute masking

At runtime PingFederate logs user attributes (see *PingFederate log files* on page 101). 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 *Manage data stores* on page 168).
- Retrieval of attributes from an IdP adapter or token processor (see *Set pseudonym and masking options* on page 324 and *Set attribute masking* on page 475).
- SP-server processing of incoming attributes based on the SSO attribute contract, (see *Define an attribute contract* on page 418).

  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 78).
- SP-server processing of incoming attributes in response to an Attribute Request under XASP (see *Configure security policy for Attribute Query* on page 434).

  For information about XASP, see *Attribute Query and XASP* on page 47.

  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 (see *PingFederate log files* on page 101).

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 347, 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.

Related concept
Attribute mapping expressions on page 593

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 85).

  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 85).

Outbound provisioning for IdPs

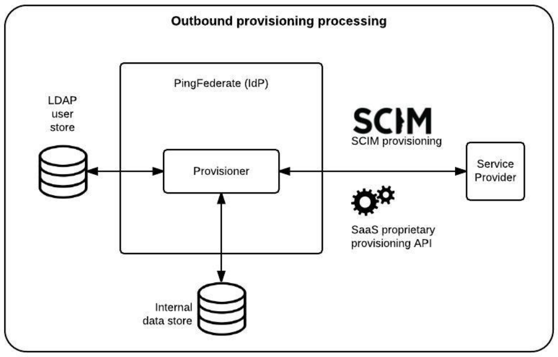
User provisioning is an important aspect of identity federation. Often when organizations enable SSO for their users, they must ensure that some form of account synchronization is in place. Automated user provisioning features within PingFederate free administrators from having to devise a manual strategy 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 441. To integrate inbound provisioning with custom user stores, see *Configure Identity Store Provisioners* on page 403. For application-development information about using PingFederate endpoints for SCIM provisioning, see *SCIM inbound provisioning endpoints* on page 536.

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 412. For configuration information, see *Configure just-in-time provisioning* on page 434.

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:

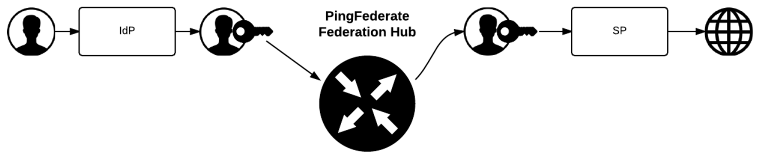
- *Bridging an IdP to an SP* on page 87
- *Bridging an IdP to multiple SPs* on page 87
- *Bridging multiple IdPs to an SP* on page 88
- *Bridging multiple IdPs to multiple SPs* on page 89

PingFederate also supports protocol translation among SAML 1.0, 1.1, 2.0, OpenID Connect (for identity providers), 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

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.

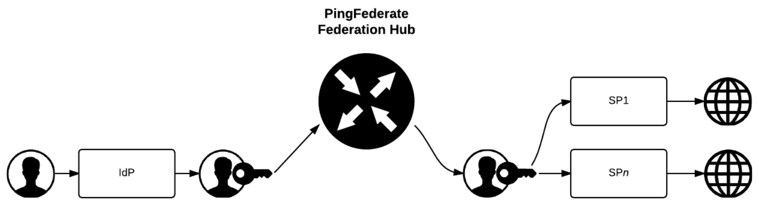


To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols on the **Server Configuration** > **Server 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 90).
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

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.



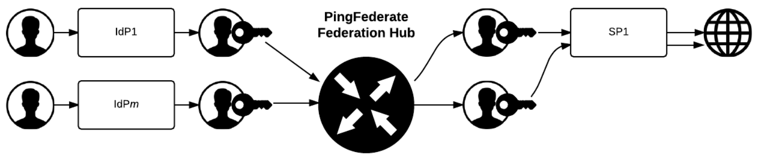
To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols on the **Server Configuration** > **Server 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 90). 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

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.



To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols on the **Server Configuration** > **Server 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 90). 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 to map each identity provider to the corresponding IdP connection.
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.

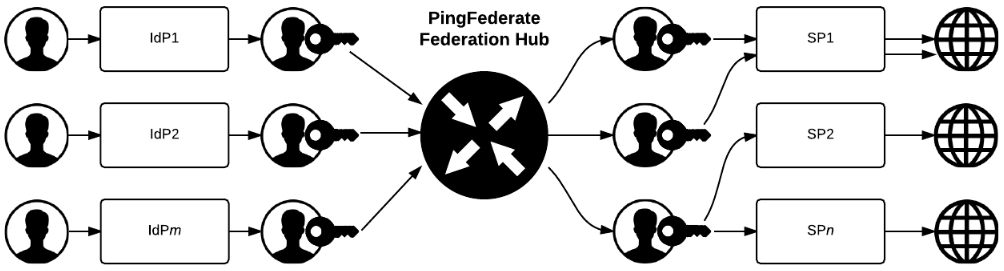
    For information about Authenticating Authority, see section *2.7.2.2 Element <AuthnContext>* in the SAML 2.0 specification (https://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf)

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

This PingFederate federation hub use case is a combination of *Bridging an IdP to multiple SPs* on page 87 and *Bridging multiple IdPs to an SP* on page 88.



To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols on the **Server Configuration** > **Server 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 90).
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 to map each identity provider to the corresponding IdP connection.
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.

Tip: Extensive coordination and configuration may be avoided by using Auto-Connect™ (see *Auto-Connect* on page 94).

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 74.)

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 75).

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 632*).

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 93).

With PingFederate, you define a unique ID for each supported protocol (see *Specify federation information* on page 162). 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 92).

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 *Configure virtual host names* on page 121).

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: LDAP, JDBC, or Custom (see *Data stores* on page 81).

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 72 and *Token processors and generators* on page 63.)

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 101.)

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 78).

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 79.)

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. (If your partner is also using PingFederate or supports standards permitting runtime metadata exchange, the process can be even simpler—see *Auto-Connect* on page 94.

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 *Export metadata to an XML file* on page 122 and *System-services endpoints* on page 542 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`

	Partner's federation ID	Your virtual server ID
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 336) or as an SP configuring an IdP connection (see *Identify the partner* on page 413) 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 528 as an IdP or *SP endpoints* on page 532 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 344, *Restrict a target session to certain virtual server IDs* on page 420 and *Restrict a target session to certain virtual server IDs* on page 420).

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 479 or *Restrict a token generator to certain virtual server IDs* on page 486).

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 347 or *Define issuance criteria for SP Browser SSO* on page 422.

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 79).
- **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.

Auto-Connect

PingFederate allows organizations to provide secure SSO on the fly—that is, without the need for configuring partner-specific, browser-based SSO connection parameters. This feature—Auto-Connect™—extends SAML 2.0 SP-initiated SSO or SLO and metadata specifications to enable deployments to retrieve partner connection information securely on an as-needed basis. (For information about SAML 2.0, see *Supported Standards*.)

The feature is especially useful to an SP who wants to provide SSO capability to more than one partner. A SaaS provider, for example, can provide SSO to innumerable clients without specifying redundant connection information for each one. Auto-Connect can also help an enterprise, acting as an IdP, to provide easily scalable SSO for multiple outsourced services.

For either an IdP or SP PingFederate server, you can implement Auto-Connect for any number of partners by configuring a common initial setup and a list of domain names. For an IdP, the domain-name list contains SP partners from whom your site will accept Auto-Connect authentication requests. For an SP, the list contains IdP-partner domains to which your site can send authentication requests and receive SSO assertions.

For information about configuring Auto-Connect for your federation partners, see *Configure SP Auto-Connect* on page 378 or *Configure IdP Auto-Connect* on page 468.

Providing metadata

You enable Auto-Connect™ on the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen (see *Choose roles and protocols* on page 161). Once Auto-Connect is enabled and the initial setup is fully configured and activated, partners can retrieve your connection metadata via HTTP. At runtime, Auto-Connect deployments at partner sites use the endpoints provided in the metadata to interact with your server and complete SSO or SLO processing.

The metadata, which follows SAML 2.0 specifications, must be signed, and the validity of the data is time-limited (see *Auto-Connect security model* on page 95 and *Configure metadata lifetime* on page 166).

Runtime processing

Auto-Connect™ runtime processing starts when a user tries to reach a protected SP resource. The process depends on SP Web-application functionality that determines the user's IdP domain (for example, from a submitted email address) and passes it to the SP PingFederate server in the SSO request.

This illustration and the accompanying processing steps describe the complete SSO processing flow:

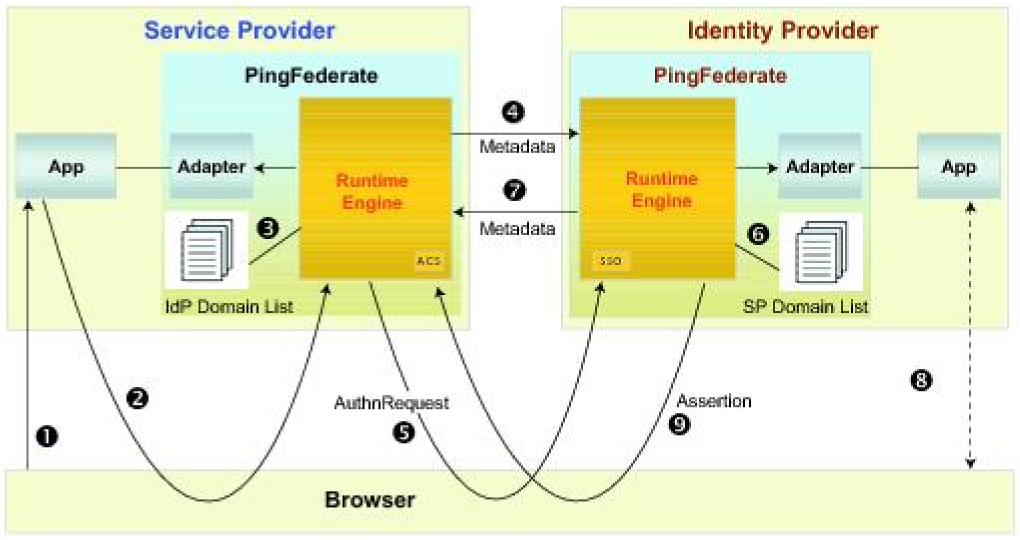


Figure 20: Auto-Connect Processing Flow

Processing steps

1. User sends a logon request with an email address to an SP application. For example:

   user@example.com
2. The application parses the email address and sends a request to PingFederate. For example:

   https://www.example.org:9031/sp/startSSO.ping/?Domain=example.com
3. The SP PingFederate server looks up the domain in a list of domain names allowed to use Auto-Connect.
4. If the domain is in the list, the SP retrieves connection metadata from the IdP's public endpoint.

   By default, PingFederate looks for the metadata by prepending `http://saml` to the domain. For example:

   http://saml.example.com

   This default location can be changed, if necessary, in the Allowed Domains lists configured in the PingFederate administrative console.
5. After validating the metadata (see *Auto-Connect security model* on page 95), the SP sends an authentication request to the IdP's SSO service.
6. If the request `<Issuer>` is not among the IdP's static-connection partners, the IdP PingFederate server looks for the issuer's domain name in the list of domains allowed to use Auto-Connect.
7. The IdP retrieves the SP's metadata via its public endpoint and verifies the metadata signature.

   The process is the same as that used by the SP in *Step 4*.
8. The IdP requests user authentication via the configured adapter instance.
9. Once the user is authenticated, the IdP returns a signed SAML assertion to the SP's Assertion Consumer Service (ACS) endpoint.
10. (Not shown) The SP logs the user on to the requested resource via the configured SP adapter.

Auto-Connect security model

Auto-Connect™ processing requires digital signatures to ensure the authenticity of the published metadata as well as all subsequent SSO or SLO requests and responses. The certificate used to sign the metadata is included in the metadata, and all certificates must be signed by a trusted Certificate Authority; thus, partners need not exchange certificates out of band.

In addition to validating certificates, the PingFederate runtime server compares the partner certificate with the entity ID (the “Issuer”) found in the SAML message. Then the server matches the entity ID against the configured list of allowed Auto-Connect domains.

This diagram illustrates the security validation process:

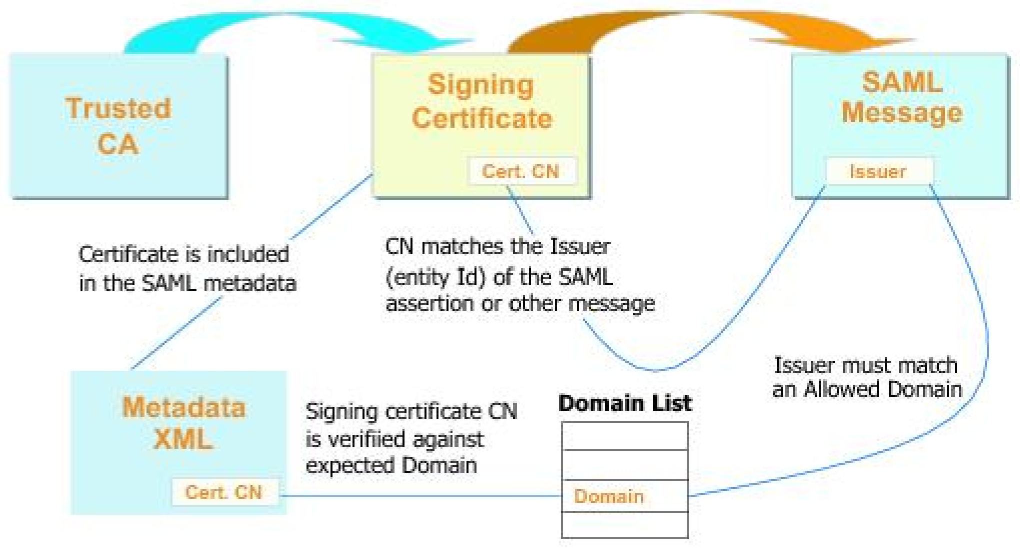

Trusted CA
Signing Certificate
Cert. CN
SAML Message
Issuer
Certificate is included in the SAML metadata
CN matches the Issuer (entity Id) of the SAML assertion or other message
Issuer must match an Allowed Domain
Metadata XML
Cert. CN
Signing certificate CN is verified against expected Domain
Domain List
Domain

Figure 21: Auto-Connect Security Model

Note that the diagram assumes that the same certificate is used for signing both the metadata and the runtime SAML messages. This is convenient, but not required.

System administration

This chapter describes general administrative functions for PingFederate, including:.

- *PingFederate log files* on page 101
- *Export metadata to an XML file* on page 122
- *Sign XML files* on page 124
- *Configuration archive* on page 124
- *Account management* on page 114
- *Alternative console authentication* on page 118
- *Manage email configuration* on page 120
- *Configure virtual host names* on page 121
- *Configure PingFederate properties* on page 98
- *Manage PingFederate license* on page 96
- *Automating configuration migration* on page 126
- *Outbound provisioning CLI* on page 130

Note: The information in this chapter is presented from the viewpoint of an administrative user with “Admin” permissions (see *Account management* on page 114).

Manage PingFederate license

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.

To review the existing license key:

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** in the drop-down menu.

The license summary is displayed in a pop-up browser window.

Note that if an expiration date is specified, the license expires at the beginning of the day.

Request a new license key

1. Go to the Ping Identity *licensing* website.
2. Sign on, provide the required information, and then submit your request.

You shall hear from our licensing team as the team processes your request.

Install a license key on a new or upgraded PingFederate server

1. Start PingFederate and sign on to the administrative console.
2. On the **Import License** 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 **Server Configuration** > **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

You may replace your existing PingFederate license key by importing a new license file in the **Server Configuration** > **License Management** screen.

1. Start PingFederate and sign on to the administrative console.
2. Go to the **Server Configuration** > **License Management** screen.
3. On the **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.

4. If you have a clustered PingFederate environment, go to the **Server Configuration** > **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 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.

Configure notification for licensing events

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.

1. Go to the **Server Configuration** > **Server Settings** > **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, and then click **Save**.

   If you have not yet configured email server settings, click **Email Server Settings** to provide the required information. When you complete the email server configuration, the administrative console brings back the **Runtime Notifications** screen. Click **Save**.

Configure 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. 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 647.

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.
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 example, with SOAP or artifact SAML bindings or for WS-Trust STS calls. To use this port, change the placeholder value to the port number you want to use. **Important:** If you are using mutual SSL/TLS for either WS-Trust STS authentication or for SAML back-channel authentication, you *must use* this port for security reasons (or use 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.

Property	**Description**
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`
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 server (as defined on the **Server Configuration** > **Server Settings** > **Federation Info** screen).
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 208.
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 153).
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.)

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.

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 610.

Updating the log level for existing `Logger` elements does not require a restart. The adjustments are activated within half a minute. Other changes require a restart of PingFederate; for example:

- Adding new `Logger` elements
- Updating the size attribute in the `RollingFile/Policies/SizeBasedTriggeringPolicy` element
- Updating the max attribute in the `RollingFile/DefaultRolloverStrategy` element
- Updating the filePattern attribute without modifying the fileName attribute in the `RollingFile` appender

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 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

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)

Mode	Description
	* 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.

- 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 elements. PingFederate separates each element by a vertical pipe (`|`). As needed, elements are configurable by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

Default elements (in the order that PingFederate records them in the security audit log):

Element	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`, and `AUTHN_REQUEST`. `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.
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 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`.

Element	**Description**
host	The host name or IP address of the PingFederate server.
role	The role PingFederate played for the transaction.
status	The status of the transactions.
adapterid	The ID of an adapter instance. Consider adding the authenticationsourceid and targetsessionid elements to log 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 elements (in alphabetical order):

Element	**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 or IdP connections (or both) invoked in an authentication or logout flow; for example, `[idpAdapterOne, idpConnectionX]`
granttype	The OAuth grant type.
header{*anHttpRequestHeader*}	The HTTP request header value identified by the header name. The header name is case-insensitive; for example, `header{user-agent}` and `header{User-Agent}` are equivalent. To record multiple headers, repeat the header entry, 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.

Element	Description
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).
outxmlmsg	The outgoing message; for example, a SAML Response or the information pertaining to a response for an OAuth request.
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.
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.
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 or UNIX.

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.

Write 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. 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 4.1 drivers. 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.

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 or UNIX. 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.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 via HP ArcSight tools. Alternatively, administrators can write the information to a flat file in CEF; however, using syslog, when available, is recommended.

Note: PingFederate is certified with HP ArcSight for interoperability using the default elements defined in `log4j2.xml`. Any additions to these elements may render your CEF logging incompatible with HP ArcSight.

Write audit log in CEF

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.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.

Write provisioner audit log in CEF

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)
   - `OutboundProvisionerEventToCEFFile` (a `RollingFile` appender under the **Outbound provisioner audit log for CEFFile** section)

   Note: The `OutboundProvisionerEventToCEFSyslog` `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).

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.

Write audit log for Splunk

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.

Ping Identity provides a custom Splunk App for PingFederate to process audit logs generated by a PingFederate deployment. The application 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.)

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.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.

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).

Account management

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 native 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 **Server Configuration** > **Configuration Archive** screen in the administrative console
- The `/pf-mgmt-ws/ws/ConnectionMigrationMgr` and `/pf-mgmt-ws/ws/ConfigReplication` Connection Management Web Service endpoint
- The **Server Configuration** > **Application Authentication** screen in the administrative console for the **Connection Management** service

For native authentication, access and authorization are controlled by the local accounts defined in the **Server Configuration** > **Account Management** screen (unless the administration style has been updated to the **Single-User Administration**).

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 logon attempts from the same location within a short time period, the administrative console will temporarily lock out further attempts by the same user. The user must wait one minute to try again.

Note that the accounts in the **Account Management** 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 **Account Management** screen appears only if the administrative API is left to use native authentication, and vice versa.

Tip: If you have connected PingFederate to PingOne, you may also single sign-on from the PingOne admin portal to the administrative console.

Enable native authentication

When the administrative console is protected by native authentication, access is restricted to the local accounts defined in the **Server Configuration** > **Account Management** screen.

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.

Manage native accounts and role assignments

1. Go to the **Server Configuration** > **Account Management** screen.

Task	Steps
Create a native account	**1.** On the **Account Management** screen, click **Create User**. **2.** 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. **3.** 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. **4.** On the **Summary** screen, review your configuration, modify as needed, and then click **Done**. **5.** On the **Account Management** screen, select the applicable account type (**Auditor** or **Admin**) and one or more administrative roles for an **Admin** account. **6.** Repeat these steps to create additional accounts.
Modify user information	**1.** On the **Account Management** screen, select the account by its username. **Note:** Applicable only to active accounts. **2.** On the **User Information** screen, update the record, and then click **Done**. **3.** Repeat these steps to update other accounts.
Update role assignments	**1.** Select a different account type (**Auditor** or **Admin**) for one or more accounts. **2.** Select or clear the check boxes that correspond to the three administrative roles (**User Admin**, **Admin**, and **Crypto Admin** for one or more accounts. **Note:** Applicable only to the **Admin** accounts.
Deactivate or reactive a native	**1.** Click **Deactivate** or **Activate** under **Action**. **2.** Repeat this step to deactivate or reactive other accounts. **Note:** For traceability and accountability purposes, native accounts cannot be deleted; their records are retained and they can be reactivated if needed.

2. Click **Save** to keep your configuration.

Enable notification for account management events

Administrators may enable email notification for account management events.

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.

Note: If you are using an alternative console authentication for the administrative console or the administrative API, password notifications (if any) is handled by the third-party system.

1. Go to the **Server Configuration** > **Account Management** screen.
2. Select the **Notify Administrator of Account Change** check box.

   An email address must be provided for the applicable accounts.

3. If you have not yet configured PingFederate to use your email server, click **Email Server Settings** and complete the configuration.
4. Click **Save** to keep your configuration.

When enabled, email notification is triggered by the following events:

Event	Notification
An administrator has turned off the **Notify Administrator of Account Changes** option.	An email notification is sent immediately 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.	An email notification is sent immediately to the previous email address and the updated email address. The message includes the username of the administrator who made the change.
An administrator's password has been changed.	An email notification is sent immediately to the administrator whose password has been changed. The message includes the username of the administrator who made the change.

Administrators should review the changes and ensure that the changes legitimate.

Set or reset password

User administrators generate temporary passwords as they create new native 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.

1. Go to the **Server Configuration** > **Account Management** screen.
2. Optional: Select the **Notify User of Password Change** check box if you want PingFederate to notify the user about password changes via email.
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**.

   After you click **Save**, if you have enabled email notification for password changes, the new password is emailed to the user automatically.

Change password

Administrative users and auditors can change the passwords of their native 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.

1. Go to the **Server Configuration** > **Account Management** screen.
2. Click **Change Password** under **Action**.
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.

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 **Server Configuration** > **Account Management** 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 native 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 **Server Configuration** > **Configuration Archive** screen in the administrative console
- The `/pf-mgmt-ws/ws/ConnectionMigrationMgr` and `/pf-mgmt-ws/ws/ConfigReplication` Connection Management Web Service endpoint
- The **Server Configuration** > **Application Authentication** screen in the administrative console for the **Connection Management** service

Enable LDAP authentication

The LDAP authentication setup is available via configuration files in the `<pf_install>/pingfederate/bin` directory.

Note: When LDAP 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 LDAP server and enforced according to its password lockout settings.

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.

Enable RADIUS authentication

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. 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.

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.

Enable 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.

Other setup steps, including designating user permissions, are required via configuration files 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.

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 **Server Configuration** > **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.

Manage email configuration

If you are using email notifications for password resets, licensing events, certificate-expiration warnings, or automatic metadata-reloading events, you must set up and maintain a connection to the email server that PingFederate uses to send messages.

Tip: In a clustered PingFederate environment, the email notifications are only sent by one of the nodes in the cluster: the console node or any engine node. If this node leaves the cluster (planned, or not), another node picks up this task automatically.

1. Go to the **Server Configuration** > **Email Configuration** screen.

   Applicable only if you have already configured email notifications in the **Server Configuration** > **Server Settings** > **Runtime Notifications** or **Account Management** screen.

   When you configure email notifications for the first time in the **Runtime Notifications** or **Account Management** screen, the administrative console guides you to the **Email Configuration** screen to set up an email server.
2. Configure the email server settings as follows:

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 hostname of your email server.
SMTP Port (Required)	The SMTP port on your email server. The default value is `25`.
Encryption Method (Required)	Select **SSL/TLS** to establish a secure connection to the email server at the SMTPS port.

Field	Description
	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. This field is not active unless **SSL/TLS** is the chosen encryption method. The default value is `465`.
Verify Hostname	Indicates whether to verify the hostname of the email server matches the Subject (CN) or one of the Subject Alternative Names from the certificate. Visible and selected when either **SSL/TLS** or **STARTTLS** is the chosen encryption method.
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.
Retry Delay (Minutes)	The amount of time in minutes that PingFederate waits before trying to send an email notification again.
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.
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.

3. Optional: Click **Test Email Connectivity** to verify connectivity with the configured server.

   Although optional, it is recommend that you run a connectivity test. A message next to the button indicates a successful test. Verify that the test email address received a message from the server. Test reports are written to the server log, located in the `<pf_install>/pingfederate/log` directory.
4. Click **Save** to commit the email server configuration.

Configure virtual host names

In certain contexts, the SAML specifications require that XML messages include a URL identifying the host name to which the sender directed the message. (The name of the XML element containing the URL varies among protocols.) In addition, the recipient must verify that the value matches the location where the message is received.

Depending on your networking requirements, this specification can present problems; for example, in the case of proxy forwarding, where the final destination host name might be unknown to your federation partner. To provide more flexibility in such cases, you can set up a list of alternative host names on the **Server Configuration** > **Virtual Host Names** screen for PingFederate to use as part of its message-security validation.

Note that virtual host names are used for a different purpose than virtual server IDs, which provide separate unique identifiers on a per-connection basis for a federation deployment, normally in the *same* domain. Depending on your needs, however, you can configure virtual server IDs and virtual hosts in the same installation of PingFederate.

- 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.

Export metadata to an XML file

For SAML deployments, if your partners do not support consuming metadata by URL, you can export metadata for any connection or select certain non connection-specific metadata for export. Although optional, it is recommended to sign the metadata, so that partners can verify the authenticity of the metadata.

1. Go to the **Server Configuration** > **Metadata Export** screen.
2. On the **Metadata Role** screen, select the applicable role.
3. On the **Metadata Mode** screen, select the information to be included in the metadata and whether to use the SOAP channel.
    a) Select the **Use a connection for metadata generation** option if you want to export metadata for a specific connection; otherwise select the **Select information to include in metadata manually** option to create a metadata export file that is not bounded to a specific connection.
    b) Select the **Use the secondary port for SOAP channel** check box if the PingFederate secondary HTTPS port is configured and you want to use it for the SOAP channel.

    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. Follow the rest of the workflow to export a metadata XML file, including selecting the certificate to sign the metadata XML file in the **Metadata Signing** screen (as needed).
5. Pass the metadata XML file to one or more partners.

Provide SAML metadata by file

You can export metadata for any SAML connection to an XML file. This is useful in a situation, where you have already created a SAML connection to the partner and your partner prefers consuming SAML metadata by file.

1. Go to the **Server Configuration** > **Metadata Export** screen.
2. On the **Metadata Role** screen, select the applicable role.
3. On the **Metadata Mode** screen, select the information to be included in the metadata and whether to use the SOAP channel.
    a) Choose the **Use a connection for metadata generation** option.
    b) Select the **Use the secondary port for SOAP channel** check box if the PingFederate secondary HTTPS port is configured and you want to use it for the SOAP channel.

    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 connection from the list.

    Note: 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 connection metadata are specific to the selected virtual server ID. Re-export the connection metadata for your partners after updating your virtual server IDs.
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.
7. Pass the metadata XML file to your partner.

Provide general SAML metadata by file

You can manually select the desired information and generate a metadata XML file. This is useful for the following situations:

- You have not yet created a SAML 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 generate a non connection-specific metadata with selected information, which can be passed to multiple partners to expedite their configurations.

1. Go to the **Server Configuration** > **Metadata Export** screen.
2. On the **Metadata Role** screen, select the applicable role.
3. On the **Metadata Mode** screen, select the information to be included in the metadata and whether to use the SOAP channel.
   a) Choose the **Select information to include in metadata manually** option.
   b) Select the **Use the secondary port for SOAP channel** check box if the PingFederate secondary HTTPS port is configured and you want to use it for the SOAP channel.

      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 applicable federation protocol that supports metadata exchange from the list.

   If you have only enabled one protocol that supports metadata exchange in the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen, this is a read-only screen with such protocol.
5. Optional: On the **Attribute Contract** screen, add one or more attributes.

   Click **Edit**, **Update**, **Cancel**, or **Delete** under **Action** to modify or remove any attributes.
6. Optional: Select a signing key from the list.

   When selected, the metadata export contains the public key that your partners can use to verify the digital signature of your SAML messages.
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 to SAML 2.0.

   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.
10. Pass the metadata XML file to one or more partners.

Sign XML files

PingFederate supports digital signing of SAML metadata files that you and your partner might want to exchange. Applying a digital signature to an XML file ensures that the file is from the original source and that its contents have not been modified by a third party.

If you have previously exported an unsigned metadata XML file, you can create a (new) signed metadata XML file based on the unsigned metadata.

1. Go to the **Server Configuration** > **XML File Signatures** screen.
2. On the **Select XML File** screen, choose your file.
3. On the **Digital Signature Settings** screen, choose the certificate containing your signing key from the list.

   Note: By default, certificate and public-key information is included in the signed XML file. If you do not wish to include this information, clear the "**... <KeyInfo> ...** " and "**... <KeyValue> ...**" check boxes.
4. Select a signing algorithm corresponding to the selected certificate. Choices include RSA-SHA1, SHA256, SHA384, and SHA512; ECDSA-SHA256, SHA384 and SHA512; and SHA1.
5. On the **Export & Summary** screen, click the **Export** button to save the digitally signed file.

Replicate configuration

From the **Server Configuration** > **Cluster Management** screen, which is available only when the administrative console is running in cluster mode, you can replicate the current console configuration to all server nodes in the cluster.

This screen is also useful for verifying that nodes have joined the cluster.

For more information, refer to *Console configuration push* on page 654 in the PingFederate *Server Clustering Guide on page 632*.

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 **Server Configuration** > **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 **Server Configuration** > **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.

Configure a backup schedule

For deployments with hundreds of connections or OAuth clients (or both), administrators can optionally configure PingFederate to create configuration archives periodically instead.

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. 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.

Export an archive

Administrators can export the current administrative-console configuration to a ZIP file on the **Server Configuration** > **Configuration Archive** screen. This screen is only available to administrators, whose accounts have been assigned the User Admin, Admin, and Crypto Admin roles.

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.

Import an archive

Administrators can import a configuration archive (ZIP) on the **Server Configuration** > **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 **Server Configuration** > **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.

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**.

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 Web 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 application authentication*).

Caution: For security reasons, the Connection Management Service should be disabled whenever it is not in use.

Copy 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. 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.

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

Migration tool

The `configcopy` tool may 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.

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 UNIX/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 Web Services are 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.

1. Ensure access to the Connection Management Web Service is enabled for both the source and target PingFederate servers (see *Configure application authentication*).
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 management* on page 192).

   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 Server JRE 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 Server JRE 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

Command argument	Description
--show-nodes	Shows all the provisioning-server nodes with their status and the last timestamp (applicable only when failover provisioning is configured in the `<pf_install>/pingfederate/bin/run.properties` file).
--force-node-backup Use with node number: -n <node ID>	Sets the provisioner mode to FAILOVER for the associated PingFederate server node.

Channel-specific options

Note: With each command, specify the channel with the `-c <channel-id-number>` argument; for example:

```
provmgr -c 1 --show-source
```

You can determine channel ID numbers by using the global command: `provmgr --show-channels`

--reset-group-timestamp	Deletes the user-group timestamp, which forces the provisioner to process the provisioning group on the next cycle, even if the timestamp on that group did not actually change. Depending on your LDAP server and administrative practices, you 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.
--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:	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.

Command argument	Description

-u <provider name>
Or:
-g <LDAP GUID>
	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 **Server Configuration** > **Server 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

Variable	Description
	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.
`$PingFedBaseURL`	The PingFederate base URL, as defined on the **Server Configuration** > **Server Settings** > **Federation Info** screen.
`$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 *html.form.login.template.html*	Displays a configurable 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 his or her password via the HTML Form Adapter.	Normal	User input required
Change Password *html.form.message.template.html*	Displayed when a user has successfully changed his or her password.	Normal	User input required
Password Expiring *html.form.password.expiring.notification.template.html*	Displayed to warn an authenticated user that the password associated with the account is about to expire.	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 his or her password.	Normal	User input required
Account Recovery *forgot-password.html*	Displayed when a user attempts to reset his or her 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 his or her 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 an email or text message notification or to notify a user to refer to the email 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**	Normal	User input required

Page title and *template file name*	**Purpose**	**Type**	**Action**
	for the invoked HTML Form Adapter instance.		
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 his or her 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 his or her account through the HTML Form Adapter. This page also prompts the user to retain the current password or reset it.	Normal	User input required
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

Page title and *template file name*	Purpose	Type	Action
Manage Your Profile *local.identity.profile.html*	Displayed when an authenticated user accesses the profile management endpoint. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that has been configured to support self-service profile management.	Normal	User input required
Email Verification *local.identity.email.verification.sent.html*	Displays a notification that an email ownership verification message has been sent when an authenticated user accesses the email ownership verification endpoint. Applicable only if the invoked HTML Form Adapter instance is associated with a local identity profile that 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.

Page title and *template file name*	Purpose	Type	Action
			Alternatively, the user can contact their IT administrators for further assistance.
Username Recovery *username.recovery.template.html*	Displays to prompt the user to enter an email address to recover the username associated with the account. Applicable only if the invoked HTML Form Adapter instance is configured to support self-service username recovery.	Normal	User input required
Username Recovery *username.recovery.info.template.html*	Displays to notify the user to retrieve the email 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 email 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

Page title and *template file name*	Purpose	Type	Action
Sign Off Successful *idp.slo.success.page.template.html*	Displayed when an SLO request succeeds and no other SLO success landing page is specified.	Normal	None
Sign Off Error *idp.slo.error.page.template.html*	Displayed when an SLO request fails and no other SLO error landing page is specified.	Error	User should close the browser

WS-Federation and OpenID Connect

Page title and *template file name*	Purpose	Type	Action
Working . . . *sourceid-wsfed-http-post-template.html*	Used to auto-submit a WS-Federation assertion to the SP. If JavaScript is disabled, the user is prompted to click a button to POST the assertion directly. 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	Normal	None

Page title and *template file name*	Purpose	Type	Action
	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.		
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 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 two screens presented to end users (resource owners) during OAuth transactions, one requesting approval of the scope of protected resources requested and one providing a means of revoking persistent access grants. 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

Page title and *template file name*	Purpose	Message type	Action
Request for Approval *oauth.approval.page.template.html*	Advises resource owners that their information is being requested by the identified OAuth client and provides for approval/disapproval. This page appears for Implicit or Authorization Code grant types, either one time only 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.	Normal	User input required

Customizable email notifications

PingFederate supplies HTML templates to notify the administrators and the end users for various events. These template files are located in the `<pf_install>/pingfederate/server/default/conf/template/mail-notifications` directory.

You can modify these email notification template files in a text editor to suit the particular branding and informational needs of your PingFederate installation. Each template contains variables and standard HTML. At runtime, PingFederate supplies values for the variables and generates the email messages from the template files.

Note: If you have a clustered PingFederate environment, copy the customized templates to each node.

Email notification is optional. Unless otherwise stated, all events and their respective options, including the recipient information, are configurable on the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen. In addition, email configuration must be completed on the **Runtime Notifications** > **Email Server Settings** screen.

Account management events

Email notification for account management events is configurable on the **Server Configuration** > **Account Management** 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.

Important: In order for an administrator to receive emails about account management events, an email address must be provided on the **Account Management** > **User Information** screen when creating or modifying the administrator's profile.

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. An email notification is sent immediately 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. An email notification is sent immediately to the previous email address and the updated 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. An email notification is sent immediately 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 send email notification for certificate events, for any self-signed certificates that the administrators have enabled automatic certificate rotation on the **Server Configuration** > **Signing & Decryption Keys & Certificates** screen, PingFederate also sends 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. The email notification is sent 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

Email subject and *template file name*	**Event**	**Action**
		configuration using the administrative console.
A PingFederate Certificate Has Expired *message-template-cert-expire.html*	A certificate expired. The email notification is sent 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. The email notification is sent when a new pending certificate is created. The message includes the details of the current certificate, the details of the new certificate, the activation date, and the connections that will be affected when the new certificate is activated.	Work with the applicable partners to update the expiring certificate. PingFederate supports providing metadata for Browser SSO connections. For a clustered PingFederate environment, replicate the configuration using the administrative console.
A PingFederate Certificate Has Been Updated *message-template-cert-deactivation.html*	A new certificate for signing or decryption is activated. The email notification is sent 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.

HTML Form Adapter events

The HTML Form Adapter supports notifying the end users when their passwords have been updated, their accounts have been unlocked, or their email addressed have been updated. Administrators can optionally customize and localize the email notifications 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.

Email subject and *template file name*	**Event**	**Action**
	The email notification is sent immediately.	
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**. The email notification is sent immediately.	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**. The email notification is sent immediately.	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**). The email notification is sent immediately.	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.
Password Reset Failed *message-template-forgot-password-failed.html*	An attempt to reset the end-user's password has failed. The email notification is sent immediately.	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	Users should contact their IT administrators if they have not made any attempts to update the email address associated with their accounts.

Email subject and *template file name*	**Event**	**Action**
	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. The email notification is sent immediately.	
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.

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.

Email subject and *template file name*	Event	Action
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 on the upper-right corner, then click **About**. The license summary is displayed in a pop-up browser window.

Note that 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.

Email subject and *template file name*	**Event**	**Action**
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.

Email server verification

You must complete the email configuration on the **Runtime Notifications** > **Email Server Settings** screen in order to enable email notification for any events. Although optional, it is recommend that you enter a test recipient and run a verification test on the configuration screen.

Email subject and *template file name*	**Event**	**Action**
Test Message from Your PingFederate Server *message-template-test.html*	To verify the email server settings are functional.	Ensure the verification email is received by the intended recipient. Consult log, as needed.

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.

Localize messages for end users

PingFederate support localization for three types of end-user messages: on-screen messages, email messages, and text messages (SMS). 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`

Follow these steps to localize messages for your end users.

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.

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/javase/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.

Configure password policy

PingFederate applies a configurable policy to passwords, pass phrases, and shared secrets defined by the administrators in the administrative console. 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.

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.

Manage cipher suites

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.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 import control restrictions, the standard Java Runtime Environment (JRE) distribution supports strong but not unlimited encryption. For this reason, the strongest cipher suites in the same configuration files are also commented out. To use the strongest encryption, when permissible, remove the comments from the AES 256 cipher suites and download and install the appropriate version of "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" from the *Oracle download website* (www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html).

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.

- 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 **Server Configuration** > **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 **Server Configuration** > **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.

Manage expired OAuth persistent grants

Persistent grants can result from the following OAuth use cases:

- OAuth clients using the **Refresh Token** grant type in conjunction with either the **Authorization Code** or **Resource Owner Credentials** 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.
- OAuth clients using the **Implicit** grant type *and* the **Reuse Existing Persistent Access Grants for Grant Types** check box is selected in the **OAuth Server** > **Authorization Server Settings** screen.

  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.

PingFederate automatically removes expired persistent grants and the associated attributes once a day. The cleanup process 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 process.

Note: Increasing the frequency of the cleanup process, the number of expired grants 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.

1. Optional: Adjust the frequency of the cleanup process.

   a) Edit the `config-store/timer-intervals.xml` file, located in the `<pf_install>/pingfederate/server/default/data` 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 a configuration file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

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.

   For a clustered PingFederate environment, perform these steps on the console node because the cleanup process only runs on the console node. No changes or restart of PingFederate is required on the engine nodes.

Specify the domain of the PF cookie

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.

1. Edit the `session-cookie-config.xml` file, located in the `<pf_install>/pingfederate/server/default/data/config-store` directory.
2. Add a 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 **Server Configuration** > **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

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
```

Extend the lifetime of the PF cookie

PingFederate identifies PingFederate-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 web browser to reuse them when restarted.

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 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 **Server Configuration** > **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Configure forward proxy server settings

You can configure PingFederate to send web traffic (HTTP, HTTPS, or both) that it initiates through a forward proxy server.

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

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.

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 **Server Configuration** > **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

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.

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 **Server Configuration** > **Cluster Management** screen. It is not necessary to restart PingFederate on any running engine node.

Customize the OP configuration endpoint response

The OpenID Provider (OP) configuration endpoint at `/.well-known/openid-configuration` provides configuration information for the OAuth clients to interface with PingFederate using the 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.

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

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.

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

PingFederate provides a favorite icon (favicon) for its application (such as `/idp/startSSO.ping`) and protocol endpoints (for example, `/idp/SSO.saml2`).

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

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 continues the SSO process by moving on to the next data store in the setup. If you prefer PingFederate to abort the SSO requests (the default behavior of PingFederate 8.0 and older), you can override the behavior by modifying a configuration file.

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. 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 **Server Configuration** > **Cluster Management** screen.

System settings

The System Settings links on the Server Configuration menu provide access to global settings that may apply to either an IdP or an SP configuration.

This chapter covers:

- *Server settings* on page 155
- *Manage data stores* on page 168
- *IdP discovery* on page 190
- *Configure redirect validation* on page 187

Note: The information in this chapter is presented from the viewpoint of an administrative user with "Admin" permissions (see *Account management* on page 114).

Server settings

Server settings include unique federation server identifiers, the designation of your site's federation role (SP, IdP, or both), and your enabled federation protocols (see *Supported Standards*).

Server settings also include system-administration configuration (one-user or multi-user), email notification options and setup, and a shortcut link to account management (when multi-user administration is enabled).

If you have enabled Auto-Connect™, outbound provisioning, or both, you must configure several parameters specific to those features in the System Settings task flow.

You configure many of these settings initially during the installation setup, but you can change or add to them as needed from the **Server Configuration** menu.

Note: For information about WS-Trust STS Settings, see *Configure STS authentication* on page 471.

Enter system information

On the System Info screen, you provide general information about your company.

To reach this screen

1. Click **Server Configuration**
2. Click **Server Settings** under System Settings.
3. Click **System Info** on the Summary screen.

Configure runtime notifications

Depending on your licensing agreement, your PingFederate license may have an expiration date. On the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen, when the **Notification for server licensing events** check box is selected, PingFederate sends an email warning to an email recipient (or an email group address) when your license is about to expire. Note that this option does not appear when you have a perpetual license.

You can also configure the server to send an email notification for certificate events to the same or a different recipient. When the **Notification for certificate events** is selected, PingFederate sends email notifications when a certificate used by PingFederate is about to expire or has expired. In addition, for self-signed certificate that you have enabled automatic certificate rotation, PingFederate also sends email notifications when a new key pair and a new certificate are created, and when the new certificate is activated.

Finally, you can also configure PingFederate to send email notifications when it detects changes for SAML connections that are configured to pull metadata periodically from the partners. Select the **Notification for SAML metadata update events** check box and enter an email address for the same or yet another recipient.

Tip: In a clustered PingFederate environment, the email notifications are only sent by one of the nodes in the cluster: the console node or any engine node. If this node leaves the cluster (planned, or not), another node picks up this task automatically.

Regardless of runtime notification settings, PingFederate logs license information, certificate events, and SAML metadata update events in the server log by default (see *PingFederate log files* on page 101).

To configure notifications:

1. Select the check box next to the type of notification you want, and enter an email address.

2. If you have selected the **Notification for certificate events** check box, enter a warning time period in the **Initial Warning Event** field (optional) and in the **Final Warning Event** field.

   Note: When PingFederate is configured to send notifications for certificate events, it also sends notification if a license expires.

3. If you have not previously configured PingFederate to access your email server, click **Email Server Settings** (see *Manage email configuration* on page 120).

Configure 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.

PingFederate 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 153).

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 106).

Configure SNMP monitoring

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 *Gets* and *Traps*. 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.

Gets

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 157).

Note: Some operating systems (Solaris 10, for example) may not allow the SNMP agent to bind to privileged ports: those below 1024. Consult your operating system's documentation on how to get 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 **Server Configuration** > **Server Settings** > **Runtime Reporting** screen. As needed, you may configure both Gets and Traps.

- 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.

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, the standard Server JRE client JConsole) may request runtime information from the PingFederate JMX server.

Important: Authentication is required for JMX-client access to PingFederate runtime data (see *Configure application authentication*).

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` (For Jetty connectors including the primary and secondary PingFederate runtime server ports)	`connections` – The total number of TCP connections accepted by the server. `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`).
`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`	Various attributes measuring CPU usage and memory.

MBean	Attributes
`java.lang:MemoryPool`	
`java.lang:GarbageCollection`	
`java.lang:OperatingSystem`	

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 193).

Monitor outbound provisioning

Important: Monitoring of outbound provisioning transactions using JMX has been deprecated. PingFederate now logs outbound provisioning events to `provisioner-audit.log` (see *Outbound provisioning audit logging* on page 108).

Follow these steps to re-enable JMX monitoring.

1. Edit the `<pf_install>/pingfederate/bin/run.properties` file.
2. Change the value of the provisioner.events.monitor property from `LOG` to `JMX`.
3. Edit the `<pf_install>/pingfederate/server/default/conf/jmx.remote.access` file.
4. Change the value of the default-jmx-principal property from `readonly` to `readwrite`.
5. Save all changes.
6. Restart PingFederate.

Note: Repeat these steps on each PingFederate server configured to process outbound provisioning.

When JMX monitoring for outbound provisioning is enabled, PingFederate provides an MBean called `pf.provisioning:type=saas.provisioning.events`. The MBean exposes five JMX operations, each corresponding to the Java methods described in the following table. Each method returns a CompositeData object, which allows for the retrieval of complex data without requiring application-specific code to reside with the JMX client.

Method	Description	Parameter
`viewEvents(Boolean wasSuccessful, String eventTypeStr, String fromDate, String toDate)`	Gets an array of specific events based on the given criteria. The parameters filter the data collectively; that is, they are joined logically by "and".	wasSuccessful – If `true`, returns information only on successful transactions; `false` returns information only on failed transactions; `null` returns all transactions. `eventTypeStr` – The type of event. Valid values are: `CREATE`, `UPDATE`, `DISABLE`, `ENABLE`; `null` or an empty string returns all types. `fromDate` – See the note at the end of this table.

Method	Description	Parameter
		`toDate` – See the note at the end of this table.
`eventSummaryReport(String fromDate, String toDate)`	Gets a summary of transactions counts for the given time period. Counts are provided for success, failure, and total. Each count includes a drill-down capability, providing counts by event type.	See the note at the end of this table.
`provisioningCycleSummary(String fromDate, String toDate)`	Gets a total count of provisioning cycles for the given time period. The drill-down provides information for each cycle, including success totals for users and groups added, modified, or removed in the internal tracking database; for the target data store, success and failure totals are listed for each type of transaction.	See the note at the end of this table.
`eventSummaryReportAllData()`	Gets a summary of transaction counts with no time constraints (equivalent to `eventSummaryReport` with `null` or empty strings used as parameters).	None.
`eventSummaryRollup()`	Gets a report representing an aggregate of multiple Summary Reports covering the last 0, 1, 2, 7, 30, 60, 90, 180, and 360 days.	None.

Note: Date parameters may be formatted as either `yyyy`, `yyyy-MM-dd`, or `yyyy-MM-dd HH:mm:ss`. A `null` value or an empty string for a date parameter indicates no constraint for that end of the range.

Manage PingOne settings

On the **Server Configuration** > **Server Settings** > **PingOne Settings** screen, configure various PingOne® integration settings, such as the ability to sign on to the PingFederate administrative console using the PingOne admin portal credentials and the monitoring of PingFederate server (or servers for a clustered PingFederate environment) in the PingOne admin portal.

- To toggle the ability to sign on to the administrative console using the PingOne admin portal credentials or to monitor your PingFederate servers in the PingOne admin portal, select or clear the relevant check box.
- 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 account.

  For example, you are about to set up a new SAML application on PingOne 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 or to disconnect from PingOne. In this scenario, you should upload the new configuration to PingOne so that the new PrimaryTelephone attribute becomes available when you set up the new SAML application in PingOne.
- 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 account.

For instance, you have two PingFederate environments: testing and production. The production PingFederate server is configured with a managed SP connection to PingOne. 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 or to disconnect from PingOne. In this scenario, you should disconnect the test server from PingOne so that nothing will be uploaded to your PingOne account from the test server.

- To update the authentication key that PingFederate uses to communicate with PingOne, click **Rotate Key**.

  Periodic rotation can ensure optimal security of your environment.

PingFederate also automatically rotates the signing certificate used by the managed SP connection. For more information, see *Managed SP connection to PingOne and signing certificate* on page 202.

SSO from PingOne to the PingFederate administrative console

In PingFederate 8.0 (or a more recent release), if you have selected to connect PingFederate to PingOne® during the initial setup process, the option to SSO from PingOne to the PingFederate administrative console is enabled for you.

Tip: In this scenario, the **Initial Setup** wizard does not create any local administrative login. If you decide to disable the SSO from PingOne option later, the administrative console will bring you to the **Account Management** 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 at the beginning but have subsequently created a managed SP connection to PingOne using the **Connect to PingOne** configuration wizard, you may go to the **Server Configuration** > **Server Settings** > **PingOne 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 using their PingOne credentials.

- 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 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 by PingOne to enter your 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 PingOne 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 restart PingFederate.

Monitoring of PingFederate from PingOne

After establishing a managed SP connection to PingOne®, 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 **Server Configuration** > **Server Settings** > **PingOne Settings** screen.

Update PingOne identity repository

Once a managed SP connection to PingOne® 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. When PingFederate detects such changes, the administrative console prompts you decide whether to update PingOne or to disconnect from PingOne in a banner message.

- To upload configuration changes to your PingOne account, click **Update Identity Repository**.

  Applicable if you have made changes that *should be* propagated to your PingOne account.

  For example, you are about to set up a new SAML application on PingOne 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 or to disconnect from PingOne. In this scenario, you should upload the new configuration to PingOne so that the new PrimaryTelephone attribute becomes available when you set up the new SAML application in PingOne.
- To disconnect PingFederate from your PingOne account, click **PingOne Settings**; then click **Disconnect from PingOne** on the **PingOne Settings** screen.

  Applicable if you have made changes that *should not be* propagated to your PingOne account.

  For instance, you have two PingFederate environments: testing and production. The production PingFederate server is configured with a managed SP connection to PingOne. 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 or to disconnect from PingOne. In this scenario, you should disconnect the test server from PingOne so that nothing will be uploaded to your PingOne account from the test server.

Choose roles and protocols

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.

1. Go to the **Server Configuration** > **Server 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. (For more information, refer to the *Quick Connection Guide* contained in the PingFederate SaaS Connector package for your service provider.)
3. Optional: If you are using SAML 2.0 and want to configure Auto-Connect™, select that feature for your roles.

   Note: Clearing this check box does *not* deactivate an existing Auto-Connect configuration in production. If you have already deployed Auto-Connect and wish to suspend the deployment for any reason, use the **Initial Setup** summary screens for your respective role.

4. 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.
5. Optional: If you are using PingFederate as an SP for provisioning, select the **Inbound Provisioning** check box.
6. 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 47.
7. Click **Next** (or **Save**, if you are modifying existing selections).

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

- *OAuth configuration* on page 264
- *Identity provider SSO configuration* on page 322
- *Outbound provisioning for IdPs* on page 85
- *Service provider SSO configuration* on page 400
- *Provisioning for SPs* on page 85
- *WS-Trust STS configuration* on page 470
- *IdP discovery* on page 190
- *Auto-Connect* on page 94

Specify federation information

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 Server Identification*).

You can also use a different ID for Auto-Connect™ transactions (see *Auto-Connect* on page 94).

Field descriptions

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 *Export metadata to an XML file* on page 122).
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.

Field	Description
Auto-Connect Entity ID	(Optional) If you are using Auto-Connect™, you can specify a unique ID here for Auto-Connect processing. The value must be a fully qualified URL and should match the CN of your Auto-Connect certificates (see *Auto-Connect security model* on page 95). When a value is supplied, this ID is used instead of the SAML 2.0 Entity ID in your server's Auto-Connect metadata, as well as in associated SSO/SLO requests and responses. Use this field if you have configured regular, static SAML 2.0 connections to other partners and your SAML 2.0 Entity ID is not a fully qualified URL (see *Auto-Connect* on page 94).
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.

Note: The fields available on this screen depend on the federation protocols enabled on your server (see *Choose roles and protocols* on page 161).

To reach this screen:

1. Click **Server Configuration**
2. Click **Server Settings** under System Settings.
3. Click **Federation Info** on the Summary screen.

Configure system options

The **Server Configuration** > **Server Settings** > **System Options** screen provides global settings that allow you to:

- Turn off automatic multi-connection error checking
- Define a caching interval for data-store validation
- Configure incoming proxy settings

Configure automatic connection validation

Automatic multi-connection error checking occurs whenever you access certain supporting components, such as the **Adapters** screen, the **Password Credential Validators** screen, and the **Identity Store Provisioners** screen. The intent is to verify that all configured connections have not been adversely affected by subsequent changes in the supporting components.

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. Doing so also allows you to make configuration changes without fixing dependency errors (if any).

Note: When automatic connection validation is turned off, multi-connection 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 extended attribute (from an IdP adapter instance) that has been configured as the source of the SAML_SUBJECT attribute in an SP connection, users will not be able to complete SSO requests through this SP connection until you reconfigure the connection. When you access the connection on the **Connections** screen, the administrative console displays a visual cue, indicating the error condition. To resolve the error, open the connection by selecting its name and follow the on-screen instructions to modify the configuration that requires your attention.

- On the **Server Configuration** > **Server Settings** > **System Options** screen, select the **Disable automatic connection validation** check box if you do not want PingFederate to validate connection settings automatically.

  (This check box is not selected by default.)

Note: This option does not affect the validation of connections as they are being configured or modified. Moreover, individual connections are always validated automatically when you access them on the **Connections** screen, regardless of the setting on this screen.

Specify data-store validation intervals

Automatic data-store validation tests occur by default for any adapter-to-adapter mappings or partner connections. This validation test occurs at various points within the administrative console.

The **Data Store Validation Interval (secs)** field defines the length of time for which a successful data-store validation is cached. (Only successful test results are cached.) The data-store connection is validated using the cached result without performing the test, speeding up the validation process.

- On the **Server Configuration** > **Server Settings** > **System Options** screen, override the default value, as needed.

  The default interval is five minutes (300 seconds. A value of 0 turns off the caching and validation tests are executed with each access.

Configure proxy options

When PingFederate is deployed behind a proxy server or load-balancer, the options described in the following four sections enable PingFederate to use information in HTTP headers added by the proxy server to construct correct responses. These options, configurable on the **Server Configuration** > **Server Settings** > **System Options** 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 in the **System Options** 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 proxy server are made over HTTPS, even if HTTP is used between the proxy server and PingFederate.

Configure outbound provisioning settings

On the **Server Configuration** > **Server 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 **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

1. Select a data store from the 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).

Configure metadata signing

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.

1. Go to the **Server Configuration** > **Server Settings** > **Metadata Signing** screen.
2. Select a certificate from the **Signing Certificate** list.

   Important: If you use Auto-Connect™, the certificate CN must match the domain name associated with the **Auto-Connect Entity ID** field in the **Server Configuration** > **Server Settings** > **Federation Info** 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.
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.

   Note: For Auto-Connect, the metadata signing certificate must be trusted by your partner.

To specify a certificate:

1. Select the 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 198).
2. 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 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.

   Note: For Auto-Connect™, the metadata signing certificate must be trusted by your partner (see *Auto-Connect security model* on page 95).

Configure metadata lifetime

PingFederate provides metadata for SAML and WS-Federation connections and supports auto-update for SAML connections by reloading metadata URLs provided by the partners.

Metadata publication

PingFederate includes expiration information in metadata. This metadata expiration ensures that partners always have reasonably up-to-date information about your server. You may elect to use the default time period (one day) or adjust the **Cache Duration** value on the **Server Configuration** > **Server Settings** > **Metadata Lifetime** screen.

Partners using Auto-Connect™ metadata also cache the metadata for the future requests during the lifetime of the metadata. After the lifetime has expired, the metadata is retrieved again.

Metadata consumption

PingFederate supports automatic reloading of metadata by URL for SAML connections. It checks daily by default. As needed, you can tune the frequency by modifying the **Reload Delay** value on the Metadata Lifetime screen. For more information about automatic reloading of SAML metadata by URL, see *Import SP metadata* on page 334 or *Import IdP metadata* on page 412.

Rotate system keys

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.

- To rotate the system keys, click **Rotate** and then **Save**.

Edit and save server settings

On the Server Settings Summary screen you can view, edit, and save your configuration.

- Click **Save** if you are finished with this configuration, or click any heading to make changes.

Connect to PingOne from Server Configuration

PingOne is Ping Identity's multi-tenant, identity-as-a-service (IDaaS) solution. PingOne® enables browser-based SSO and user provisioning for Identity Providers, and provides application providers with a rapid-deployment SSO capability. PingOne can be used together with PingFederate to provide a powerful solution combining the benefits of an on-premise deployment with the flexibility of a cloud solution.

To leverage this unique capability, you need a PingOne account and a *managed* SP connection from PingFederate to PingOne. If you do not have a PingOne account, sign up at *pingone.com*.

Note: A managed SP connection to PingOne is a connection created by the **Initial Setup** wizard or the **Server Configuration** > **Connect to PingOne** configuration wizard in PingFederate 8.0 (or a more recent release).

If an SP connection to PingOne was created in PingFederate 7.3 (or an earlier release), you can delete that connection and let the **Connect to PingOne** configuration wizard guide you to create a new managed SP connection to PingOne. The benefits of a managed connection include the capability to SSO from PingOne to the PingFederate administrative console, monitoring of your PingFederate servers from PingOne, monitoring of configuration changes that may impact the connection, automatic certificate rotations, and automatic metadata exchange from PingOne to PingFederate (see *Manage PingOne settings* on page 159).

1. Go to the **Server Configuration** menu.
2. Click **Connect to PingOne**.
3. On the **Identity Provider Configuration** screen:
   - Click **New Active Directory Configuration** if you need to establish a new LDAP data store to your Active Directory.
   - Click **Existing AD or Other User Store** if you have already created an LDAP data store to your Active Directory.
4. Complete the configuration.

Directory option	Process
New Active Directory Configuration	When selected, the **Connect to PingOne** configuration wizard guides you through the process of: • Connecting your Active Directory as an LDAP data store • Creating adapters and authentication selectors to authenticate end users via the Kerberos protocol or a login form based on your network topology and the browsers being used

Directory option	Process
	• Enabling users and group provisioning to PingOne • Enabling the capability to SSO from PingOne to the PingFederate administrative console
Existing AD or Other User Store	When selected, the **Connection** wizard initiates a new managed SP connection to PingOne for you. The attribute contract is automatically extended with the following six attributes: • email • fname • lname • memberOf • objectGuid • userPrincipalName Follow the step-by-step wizard to: **1.** Add one or more IdP adapter instances to the managed SP connection. **2.** Select to retrieve additional attributes from your data store as needed. **Note:** If you need to retrieve objectGUID attribute values from an LDAP data store, configure such to be handled as binary data in the LDAP data store and select **Hex** as the attribute encoding type in the connection. For more information, see *Specify LDAP binary attributes* on page 175 and *Define encoding for LDAP binary attributes* on page 608. **Tip:** If you use the HTML Form Adapter created by the **Initial Setup** wizard (in PingFederate 8.0 or a more recent release), you can map the six attributes from the adapter contract. No data store lookup is required. **3.** Fulfill the attribute contract. **4.** Configure outbound provisioning to PingOne. **5.** Save the new managed SP connection to PingOne.

Manage data stores

PingFederate can connect to data stores to retrieve user attributes for SaaS or SCIM outbound provisioning, Browser SSO, WS-Trust STS, and OAuth transactions. It can also utilize data stores for just-in-time or SCIM inbound provisioning, OAuth client records, OAuth persistent grants, and local identities.

Browser SSO and WS-Trust STS

As an IdP, you may fulfill an attribute contract that requires information beyond that which can be derived from the user's session. For example, this information may include attributes such as an email address, a job title, or any data that can be used to customize a user's experience at the SP site.

As an SP, you may use data stores to retrieve additional attributes to package with the IdP's assertion data to meet SP adapter or token generator requirements. Such attributes may be needed, for example, to establish authorization levels or to manage the local account.

OAuth

As an OAuth authorization server (AS), you may pull user attributes from data stores to fulfill an access token attribute contract. If you support OpenID Connect, you may also map user attributes from data stores to an OpenID Connect policy contract. For instance, you may want to return an email address and mobile phone number when an OAuth client connects to the UserInfo endpoint to retrieve additional attributes using an access token. Client records or persistent grants (or both) can be stored on external storage.

Outbound provisioning for IdPs

For outbound provisioning, you configure PingFederate to connect to two data stores:

- A data store from which PingFederate provisions users to the target SP.
- Another data store to monitor the state of the user store and to keep user data synchronized between the IdP and the target SP.

Inbound provisioning for SPs

For inbound provisioning, you connect to a data store to manage local user records based on inbound requests.

You can add data stores at any time. PingFederate supports a wide varierty of directory servers and database servers. Alternatively, you can create your own driver for non-JDBC or non-LDAP data stores, such as flat files or SOAP-connected databases, using the PingFederate Custom Source SDK.

1. Go to the **Server Configuration** > **Data Stores** screen.
2. On the **Manage Data Stores** screen, you can perform the following tasks:
   - To configure a new instance: click **Add New Data Store**, and then follow the configuration wizard to set up a connection to a JDBC database, an LDAP directory server, or a custom data store.
   - 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**.

Important: You must have current connectivity from PingFederate to a data store in order to create or modify the configuration. If you find that the configuration is not editable, then your connection has been lost due to a system problem not related to the PingFederate server. The problem must be identified and corrected before you can continue.

Configure a JDBC database connection

You configure access to a database by providing basic JDBC information.

Note: PingFederate has been tested with vendor-specific JDBC 4.1 drivers. 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.

1. Go to the **Server Configuration** > **Data Stores** screen.

Task	Steps
Configure a new data store	1. On the **Manage Data Stores** screen, click **Add New Data Store**. 2. On the **Data Store Type** screen, select **Database**.
Modify an existing data store	1. On the **Manage Data Stores** screen, click the applicable data store by its name. 2. On the **Summary** screen, click **Database Config**.

2. On the **Database Config** screen, configure your JDBC connection, as described in the following table.

Field	Description
JDBC URL (Required)	The location of the JDBC database. The structure of the JDBC URL varies depending on the vendor; for example: • `jdbc:mysql://`*`databaseservername`*`/`*`databasename`* • `jdbc:oracle:thin:@`*`databaseservername`*`:`*`databasename`* • `jdbc:postgresql://`*`databaseservername`*`/`*`databasename`* • `jdbc:sqlserver://`*`databaseservername`*`;databaseName=`*`databasename`*

Field	Description
	where *databaseservername* is the DNS host name (or IP) of the server hosting the database, and *databasename* is the name of a database on that server. **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 (Required)	The name of the driver class used to communicate with the source database; for example: • `com.microsoft.sqlserver.jdbc.SQLServerDriver` • `com.mysql.jdbc.Driver` • `oracle.jdbc.OracleDriver` • `org.postgresql.Driver` The driver class name should be supplied by the database software vendor in a JAR file.
Username (Required)	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 PingFederate at runtime 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 from this data store should be masked in PingFederate log files.
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.

3. Optional: Click **Advanced** to configure additional settings.

a) On the **Advanced Database Options** screen, click **Apply Defaults** to view or restore default values.

b) Configure advanced settings, as described in 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.

Field	Description
	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.
Maximum Pool Size	The largest number of database connections that can remain in the pool for the given data store.
Blocking Timeout (ms)	The amount of time a request waits to get a connection from the connection pool before it fails.
Idle Timeout (min)	The length of time the connection can be idle in the pool before it is closed. Note that PingFederate maintains the minimum connection pool for the given data store once the pool is established.

4. Click **Save** to retain the configuration.

Configure an LDAP connection

Configure an LDAP connection to access user attributes stored in a directory server.

1. Go to the **Server Configuration** > **Data Stores** screen.

Task	Steps
Configure a new data store	1. On the **Manage Data Stores** screen, click **Add New Data Store**. 2. On the **Data Store Type** screen, select **LDAP**.
Modify an existing data store	1. On the **Manage Data Stores** screen, click the applicable data store by its name. 2. On the **Summary** screen, click **LDAP Configuration**.

2. On the **LDAP Configuration** screen, configure an LDAP connection, as described in the following table.

Field	Description
Hostname(s) (Required)	The DNS name or IP address of the directory server. 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`.
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.

Field	Description
	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.
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:** If you choose an anonymous binding, ensure that this access level provides permission to search the directory for user-account information. 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. 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

Field	Description
	intend to enable self-service password reset in any HTML Form Adapter instances that use this data store. For more information, see *Configure proxied authorization* on page 173.
Password	The password credential required to access the directory server.
Mask Values in Log	Determines whether all attribute values returned through this LDAP data store should be masked in PingFederate log files. This check box is not selected by default.

3. Optional: Click **Advanced** to configure additional settings in the **Advanced LDAP Options** screen.
4. Click **Save** to retain the configuration.

Configure proxied authorization

When connecting to PingDirectory™ or Oracle Directory Server, configure proxied authorization for the service account on the directory server if you intend to enable self-service password reset in any HTML Form Adapter instances that use this 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 LDAP service account.

- For PingDirectory, see the *PingDirectory Administration Guide* and search for *Proxied Authorization*.
- 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*.

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*.

Set advanced LDAP options

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.

1. Optional: Click **Apply Defaults** to view or restore default values.

   Tip: The default values are conservative based on the thread-pooling limits set (under `Server Thread Pool`) in the `jetty-runtime.xml` file, located in the `<pf_install>/pingfederate/etc` directory.

   If any changes are made to thread pooling, we recommend updating settings as outlined in the following table.
2. Configure advanced settings, as described in 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.

Field	**Description**
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 LDAP server matches the Subject (CN) or one of the Subject Alternative Names 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 the **Tip** in step 1). 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.
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 the **Tip** in step 1).
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).
Time Between Eviction (Milli) (Required)	The frequency in milliseconds that the evictor cleans up the connections in the pool. A value of `-1` disables the evictor.
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.
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.
DNS TTL (Milli)	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.

Field	Description
LDAP DNS SRV Record prefix	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	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 retain the configuration.

Specify LDAP binary attributes

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.

1. On the **LDAP Binary Attributes** screen, add, update, or remove binary attributes.
2. Click **Save** to retain the configuration.

Define a custom LDAP type for outbound provisioning

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.

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.

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.

Configure a custom data store

Developers can use the PingFederate Custom Source SDK to create specific drivers for non-JDBC/LDAP data stores (or more sophisticated JDBC/LDAP queries) including, for example, flat files or SOAP-connected databases. Furthermore, custom 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 installed, you can select it on the **Custom Data Store Type** screen.

1. Implement your custom data store, copy the data-store driver (JAR) file to the `<pf_install>/pingfederate/server/default/deploy` directory, and then restart the PingFederate service.

2. Go to the **Server Configuration** > **Data Stores** screen.

Task	Steps
Configure a new data store	1. On the **Manage Data Stores** screen, click **Add New Data Store**. 2. On the **Data Store Type** screen, select **Custom**.
Modify an existing data store	1. On the **Manage Data Stores** screen, click the applicable data store by its name. 2. On the **Summary** screen, click the applicable screen to modify the settings.

3. On the **Customer Data Store Type** screen, configure the instance of your custom data store as follows:
   a) Enter a unique name in the **Data Store Instance Name** field.

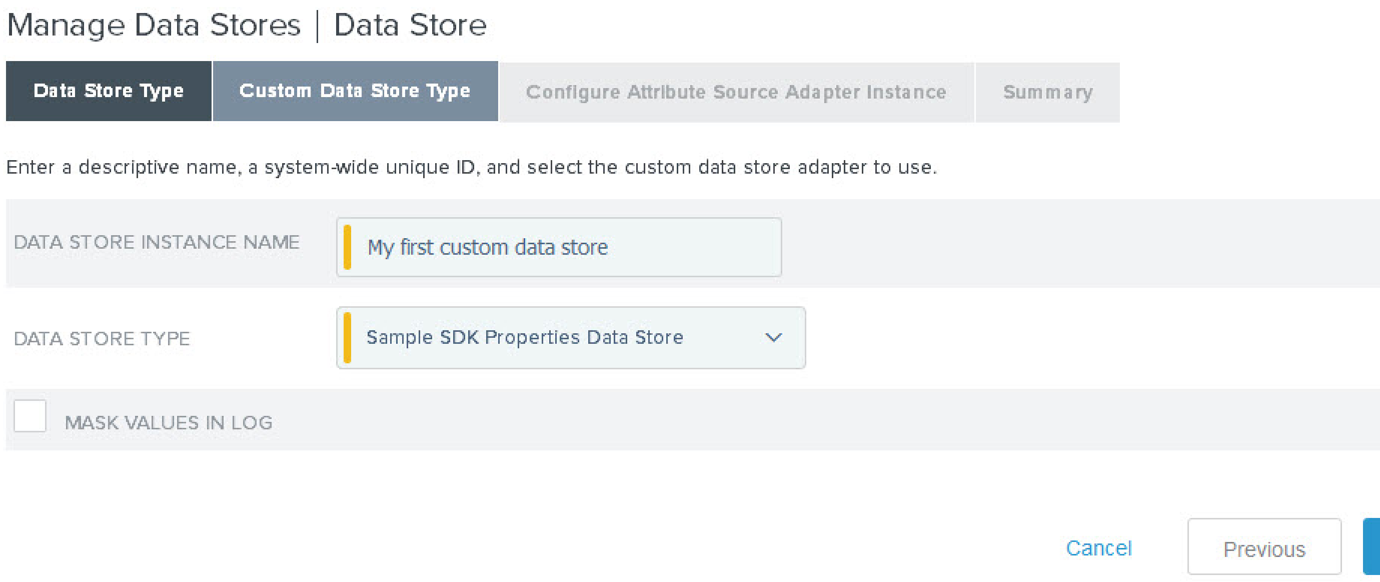
   You can create more than one instance of the same custom data store for use with different partners, as needed.

   b) Select the desired custom data store type from the list.

   If the list is empty, verify that the data-store driver is implemented and installed successfully.

   c) Select the **Mask Values in Log** check box if all attribute values returned from this data store should be masked in PingFederate log files.

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**:


Manage Data Stores | Data Store
Data Store Type
Custom Data Store Type
Configure Attribute Source Adapter Instance
Summary
Cancel
Previous

Enter a descriptive name, a system-wide unique ID, and select the custom data store adapter to use.

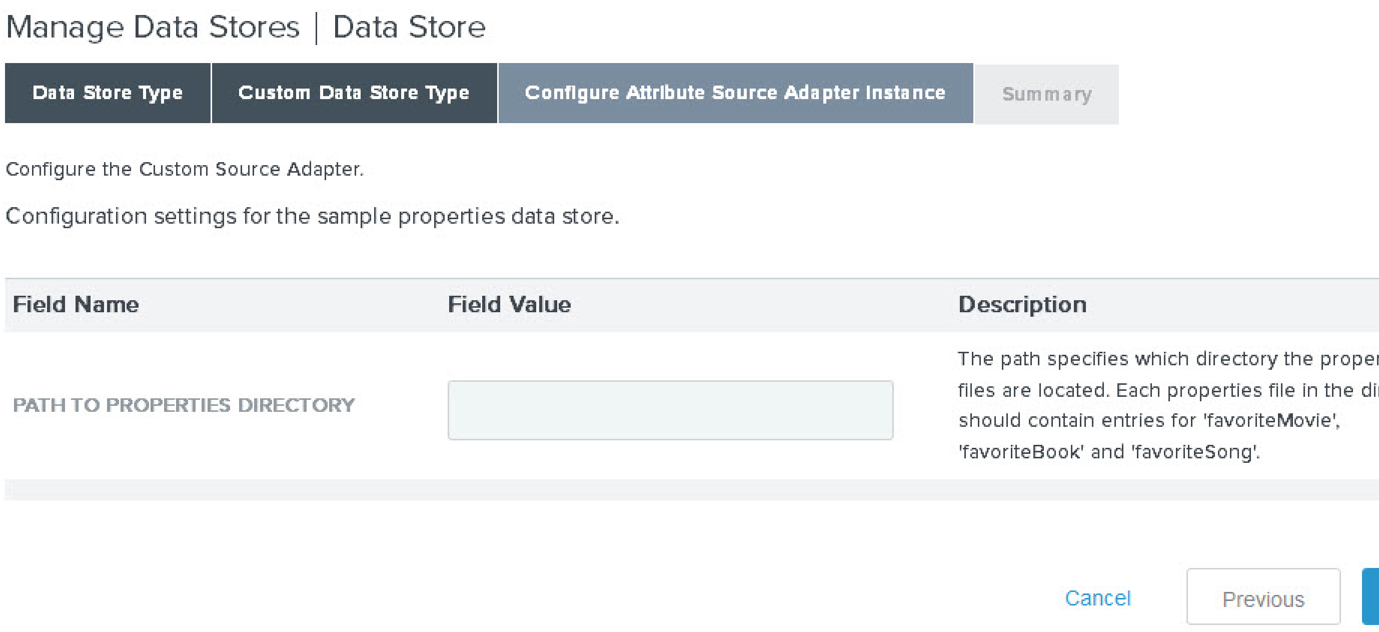
DATA STORE INSTANCE NAME: My first custom data store

DATA STORE TYPE: Sample SDK Properties Data Store

☐ MASK VALUES IN LOG

4. On the **Configure Attribute Source Adapter Instance** screen, configure the instance of your custom data store.

   This screen varies depending on the implementation. For example, the following screenshot is based on the sample from the `<pf_instal>/pingfederate/sdk/plugin-src/custom-data-store-example` directory.


Manage Data Stores | Data Store
Data Store Type
Custom Data Store Type
Configure Attribute Source Adapter Instance
Summary

Configure the Custom Source Adapter.

Configuration settings for the sample properties data store.

Field Name	Field Value	Description
PATH TO PROPERTIES DIRECTORY		The path specifies which directory the proper files are located. Each properties file in the di should contain entries for 'favoriteMovie', 'favoriteBook' and 'favoriteSong'.

5. Click the related link on the **Actions** screen to invoke an action (when applicable).
6. Click **Save** to retain the configuration.

Define an account-linking data store

When an SP is configured to use account linking for an IdP connection, PingFederate uses an embedded HSQLDB database as the account-link repository (see *Account linking* on page 78). This default implementation does not require any changes to PingFederate to support account linking. However, you can manually customize PingFederate to store account links in a different data store—either a different database or an LDAP directory. You might want to do this for any of several reasons, including:

- You are running a cluster of PingFederate runtime engines (see the PingFederate *Server Clustering Guide on page 632*). This scenario requires that you use an external database or directory for account links to ensure proper local user lookup.
- You have performance or scalability requirements that exceed the HSQLDB database's capabilities.
- You and your federation partner previously established a different system for creating and mapping opaque pseudonyms, and PingFederate needs access to the system.

Change the account-linking database

Changing the default database involves creating a table in your JDBC database to support account linking, and modifying PingFederate configuration XML files to use the database.

To create a database table for account linking:

- Run one of the table-setup scripts provided in the directory:

  ```
  <pf_install>/pingfederate/server/default/conf/account-linking/sql-scripts
  ```

If a script is not provided for your database, you can derive the setup from information available in any of the other scripts.

To change the account-linking database:

1. If you have not already done so, create a connection to the database you want to use (see *Configure a JDBC database connection* on page 169).

   Be sure to save the configuration on the **Server Configuration** > **Data Stores** screen.
2. Any time after saving the database connection, return to the **Server Configuration** > **Data Stores** screen.
3. Copy the system ID for the database you want.

4. Edit the `org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl.xml` file, located in `<pf_install>/pingfederate/server/default/data/config-store` directory.

   Between the XML tags for the `item` named `PingFederateDSJNDIName`, insert the system ID you copied at *Step 3* and save the file.
5. Start or restart PingFederate.

   Note: If you are running PingFederate in a cluster, push the new configuration to other server nodes. For more information, see the *Server Clustering Guide on page 632*.

Change the default data store to use LDAP

Changing the default data store to use LDAP involves modifying PingFederate configuration XML files to use the LDAP directory.

To use an LDAP directory:

1. If you have not already done so, create a connection to the LDAP data store you want to use (see *Configure an LDAP connection* on page 171).

   Be sure to save the configuration on the **Server Configuration** > **Data Stores** screen.
2. Any time after saving the LDAP data store connection, return to the **Server Configuration** > **Data Stores**.
3. Copy the system ID for the data store you want.
4. In the directory `<pf_install>/pingfederate/server/default/conf/META-INF`, open the file: `hivemodule.xml`

   Locate the Service-Point ID for AccountLinkingService and change the value of the `create-instance class` to:

   `org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl`
5. In the directory `<pf_install>/pingfederate/server/default/data/config-store`, open the file: `org.sourceid.saml20.service.impl.AccountLinkingServiceLDAPImpl.xml`

   Insert the following values between the XML tags for the these items:

   - `PingFederateDSJNDIName`: System ID you copied at *Step 3*
   - `UserSearchBase`: LDAP location where searches begin—for example, `CN=Users,DC=LDAPDir,DC=com`
   - `UsernameAttribute`: LDAP attribute that represents the user identifier—for example, Active Directory is `sAMAccountName`
   - `AccountLinkDataAttribute`: LDAP attribute used 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).
6. Start or restart PingFederate.
7. If you are running PingFederate in a cluster, push the new configuration to other server nodes (see the *Server Clustering Guide on page 632*)

   Important: You must manually apply the changes made in *Step 4* to the hivemodule.xml file on each server node in a cluster and then start or restart PingFederate.

   Note: User accounts to be linked must exist in the LDAP directory prior to establishing the account link. The Account Linking service does not add users to the LDAP data store but simply updates `AccountLinkDataAttribute` for a given user.

Define an OAuth grant data store

PingFederate uses its internal HSQLDB database by default to maintain persistent grants for the OAuth AS. You can also configure PingFederate to maintain persistent grants externally, on a database server, a directory server, or a custom storage medium through the PingFederate SDK.

Persistent grants can result from the following OAuth use cases:

- OAuth clients using the **Refresh Token** grant type in conjunction with either the **Authorization Code** or **Resource Owner Credentials** 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.
- OAuth clients using the **Implicit** grant type *and* the **Reuse Existing Persistent Access Grants for Grant Types** check box is selected in the **OAuth Server** > **Authorization Server Settings** screen.

  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. Note that 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.

Important: When persistent grants are expected, consider changing this configuration to maintain persistent grants securely on an external storage medium for production standalone deployments.

For server clustering, an external grant storage is required because the internal HSQLDB database cannot be shared across other PingFederate engine nodes.

Changing the default storage for OAuth persistent grants involves creating the required data structure on the external storage medium and modifying a couple PingFederate configuration XML files.

Configure an external database for grant storage

Specific tables are required in order for PingFederate to store grants, the associated attributes and their values (if any), on your database server. Table-setup scripts are provided for supported database servers.

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 to your database server on the **Server Configuration** > **Data Stores** screen.
3. Copy the system ID of the applicable JDBC data store from the **Server Configuration** > **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. Replace the `<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

   Note: For a clustered PingFederate environment, edit this file on the console node.

   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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

PingFederate automatically removes expired grants once a day. To fine-tune the frequency and the number of expired grants to be removed, see *Manage expired OAuth persistent grants* on page 150.

Configure an LDAP directory for grant storage

Specific schema objects are required in order for PingFederate to store grants, the associated attributes and their values (if any), on your LDAP directory server. LDIF scripts are provided for supported LDAP directory servers.

1. Review the LDIF scripts for your LDAP 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 LDAP 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 to your directory server on the **Server Configuration** > **Data Stores** screen.
5. Copy the system ID of the applicable LDAP data store from the **Server Configuration** > **Data Stores** screen.
6. Edit a configuration file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

LDAP 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 LDAP search base 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:

LDAP 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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

PingFederate automatically removes expired grants once a day. To fine-tune the frequency and the number of expired grants to be removed, see *Manage expired OAuth persistent grants* on page 150.

Grant storage performance considerations

If you use PingDirectory™ to store OAuth persistent grants, 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 this indexes. The `dsconfig` utility is interactive. You can also provide inputs as command arguments. For example, the following samples create the accessGrantGuid and accessGrantExpires 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 accessGrantExpires \
  --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 accessGrantGuid \
  --index accessGrantUniqueUserIdentifier \
  --index accessGrantHashedRefreshTokenValue \
  --index accessGrantClientId \
  --index accessGrantExpires
```

For more information, see the topic "Working with Indexes" in the *PingDirectory Administration Guide*.

Use a custom solution for grant storage

You may use the PingFederate SDK to implement a custom solution for grant storage.

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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Define an OAuth client data store

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, an LDAP server, or a custom storage medium through the PingFederate SDK.

Configure an external database for client storage

Specific tables are required in order for PingFederate to store OAuth client records on your database server. Table-setup scripts are provided for supported database servers.

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 *PingFederate administrative API* on page 588.

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 to your database server on the **Server Configuration** > **Data Stores** screen.
4. Copy the system ID of the applicable JDBC data store from the **Server Configuration** > **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. Replace the

`<c:item name="PingFederateDSJNDIName"/>` element value with the system ID of your data store connection.

Note: For a clustered PingFederate environment, edit this file on the console node.

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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Configure an LDAP directory for client storage

Specific schema objects are required in order for PingFederate to store OAuth client records on your LDAP directory server. LDIF scripts are provided for supported LDAP 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 *PingFederate administrative API* on page 588.

1. Review the LDIF scripts for your LDAP 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 LDAP 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 to your directory server on the **Server Configuration** > **Data Stores** screen.
5. Copy the system ID of the applicable LDAP data store from the **Server Configuration** > **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 LDAP search base 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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Client storage performance considerations

If you use PingDirectory™ to store OAuth client records, 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 this 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 the topic "Working with Indexes" in the *PingDirectory Administration Guide*.

Use custom storage for OAuth clients

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 *PingFederate administrative API* on page 588.

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 **Server Configuration** > **Cluster Management** screen; then start or restart the PingFederate service on each engine node to activate the change.

Manage CAPTCHA settings

Configure CAPTCHA settings. PingFederate supports invisible reCAPTCHA from Google.

1. Enter the site key assigned to your account by Google.
2. Enter the associated secret key.

   (There are no default values.)

Manage SMS provider settings

To connect PingFederate to Twilio as an SMS provider through which PingFederate can send text message notifications (for self-service password reset requests), enter the required information based on your Twilio account.

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).
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*.

Once saved, these SMS provider settings apply to all services using text message notifications.

Configure redirect validation

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 431)
- The default target URL for any adapter-to-adapter mappings (see *Configure a default target URL (optional)* on page 492)
- The SP default URL for successful SSO (see *Configure default URLs* on page 405)
- The IdP default URL for successful SLO (see *Configure a default URL and error message* on page 328)

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 528, *SP endpoints* on page 532 and *System-services endpoints* on page 542.

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.

1. Go to the **Server Configuration** > **Redirect Validation (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

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.

1. Go to the **Server Configuration** > **Redirect Validation** > **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**.

Related tasks
Define a service URL (WS-Federation SP connection)
Specify a service URL (WS-Federation IdP connection)

IdP discovery

IdP discovery refers to the process of identifying an end-user's IdP dynamically during an SP-initiated SSO event.

An SP can 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.

PingFederate also provides two kinds of IdP discovery:

- SAML 2.0 standard IdP Discovery
- Proprietary IdP discovery using a persistent cookie written by an SP PingFederate server

Configure IdP discovery using a persistent cookie

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.

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.

Configure 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.

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
- 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.

1. If you have not done so, go to the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen and select the IdP Discovery role.
2. Go to the **Server Configuration** > **IdP Discovery** screen, and then click **Configure IdP Discovery**.
3. On the **Domain Cookie Settings** screen, choose the discovery role or roles that PingFederate will play.

   The choices that appear on this screen depend on whether PingFederate is acting as an SP, an IdP, or both; or as an IdP Discovery server only in the **Server Configuration** > **Server 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	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	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, and then save your configuration.
7. 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.

Security management

PingFederate provides built-in certificate management to handle SSL/TLS server security, as well as certificate signing and verification of SSO and other transactions, when required.

In addition, the server provides authentication capabilities for applications making use of secured system features, or for protocol features requiring management and validation of end-user password credentials.

This section covers:

- *Certificate management* on page 192
- *Authentication* on page 210

Note: This information is presented from the viewpoint of an administrative user with “Crypto Admin” permissions (see *Account management* on page 114).

Certificate management

Using the menu items under the **Server Configuration** > **Certificate Management** section, administrators may perform the following tasks:

- *Manage trusted certificate authorities* on page 192
- *Manage SSL server certificates* on page 193
- *Manage SSL client keys and certificates* on page 196
- *Manage digital signing certificates and decryption keys* on page 198
- *Manage certificate rotation settings* on page 203
- *Manage certificates from partners* on page 204
- *Manage keys for OAuth and OpenID Connect* on page 205
- *Configure certificate revocation* on page 207

Note: Access to **Certificate Management** is restricted to administrative users with the **Crypto Admin** administrative role (see *Account management* on page 114).

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 **Server Configuration** > **Trusted CAs** screen. If the Certificate Authority 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 in `cacerts` in the Java runtime, it is not necessary to import the same certificate into the PingFederate store.

To import a certificate

1. Click **Import**.
2. On the **Import Certificate** screen:
   a) Click **Choose file** to locate and select the applicable certificate.
   b) Select the storage facility of the certificate from the **Cryptographic Provider** list.

      - **HSM**: the integrated hardware security module (HSM).
      - **Local Trust Store**: the local trust store managed by PingFederate.

      Applicable and visible only when PingFederate is integrated with an HSM from Thales in hybrid mode. (For more information, see *Supported hardware security modules* on page 56 .)
   c) Click **Next**.
3. On the **Summary** screen, review the selected certificate and then click **Save** to complete the process.

To review a certificate

1. Select the applicable certificate by its serial number.
2. Review the selected certificate in the pop-up browser window.

   When finished, close the pop-up window.
3. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To export a certificate

1. Click **Export** under **Action** for the certificate you want to export.
2. On the **Export Certificate** screen, click **Next**.
3. On the **Export & Summary** screen, click **Export** to save the certificate file on your system.

   When finished, click **Done**.
4. Click **Cancel** to go back to the **Server Configuration** screen.

To delete a certificate

1. Click **Delete** under **Action** for the certificate you want to delete.

   To undo the deletion, click **Undelete**.
2. Click **Save** to confirm the removal of the certificate.

Manage SSL server certificates

Use the **Server Configuration** > **SSL Server Certificates** screen to establish and maintain the certificates presented for access to the PingFederate administrative console and for incoming HTTPS connections at runtime.

To create a new certificate

Establish the HTTPS server certificate, which may be presented to incoming HTTPS connections.

1. Click **Create New**.
2. On the **Create Certificate** screen, enter the requested information.

   For information about each field, refer to the following table:

Field	Description
Common Name	The common name (CN) identifying the certificate.

Field	Description
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. • **HSM**: the integrated hardware security module (HSM). • **Local Trust Store**: the local trust store managed by PingFederate. Visible only when PingFederate is integrated with an HSM in hybrid mode. (For more information, see *Supported hardware security modules* on page 56 .)
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.)

Note: When using PingFederate with the Thales nShield Connect HSM, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.

Select **RSA** and an RSA signing algorithm from the **Key Algorithm** list and the **Signature Algorithm** list, respectively.

When finished, click **Next**.

3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To import a certificate and its private key

1. Click **Import**.
2. On the **Import Certificate** screen:
   a) Click **Choose file** to locate and select the applicable certificate.

      Note: When using PingFederate with the Thales nShield Connect HSM, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.

      Select a certificate that uses the RSA key algorithm.

   b) Enter the certificate password.
   c) Select the storage facility of the certificate from the **Cryptographic Provider** list.

      - **HSM**: the integrated hardware security module (HSM).
      - **Local Trust Store**: the local trust store managed by PingFederate.

      Applicable and visible only when PingFederate is integrated with an HSM in hybrid mode.
   d) Click **Next**.
3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To review a certificate

1. Select the applicable certificate by its serial number.
2. Review the selected certificate in the pop-up browser window.

   Note: If a certificate has been revoked, PingFederate indicates this problem in the certificate information window.

   When finished, close the pop-up window.
3. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To activate a certificate

1. Click **Activate for Runtime Server**, **Activate for Admin Console**, or both under **Action** for the certificate you want to activate.

   These choices are enabled only if you have created or imported more than one certificate. Otherwise, a single certificate is used for both the administrative console and runtime operations.

   To undo the activation, click **Cancel**.
2. Click **Save** to complete the process.

To create a certificate-authority signing request

1. Click **Certificate Signing** under **Action** for the desired 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.

   Tip: 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 CA signing.
2. On the **Certificate Signing** screen, select the **Generate Certificate Signing Request (CSR)** option and then click **Next**.
3. On the **Generate CSR** screen, click **Export** to save the CSR file on your system.

   When finished, click **Done**.
4. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To import a certificate-authority response

1. Click **Certificate Signing** under **Action** for the relevant certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option and then click **Next**.
3. On the **Import CSR Response** screen, click **Choose file** to locate and select the CSR response file.

   When finished, click **Next**.
4. On the **Summary** screen, review the information and then click **Save** to complete process.

To export a certificate

1. Click **Export** under **Action** for the certificate you want to export.
2. On the **Export Certificate** screen, select the type of export and then click **Next**.
   - **Certificate Only**: an export of the certificate without its private key. This is default choice.
   - **Certificate and Private Key**: an export of the certificate with its private key.

     Caution: This export contains the private key of the certificate. You must also enter an encryption password.

   If the certificate is stored on an HSM, the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file on your system.

   When finished, click **Done**.

4. Click **Cancel** to go back to the **Server Configuration** screen.

To delete a certificate

1. Click **Delete** under **Action** for the certificate you want to delete.

   This option does not appear if the certificate is in use. To enable deletion, add (if needed) and activate a different certificate for the administrative console and/or the runtime server. If the usage is not clear, click **Check Usage**.

   To undo the deletion, click **Undelete**.
2. Click **Save** to confirm the removal of the certificate.

Manage SSL client keys and certificates

On the **Server Configuration** > **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.

To create a new certificate

1. Click **Create New**.
2. On the **Create Certificate** screen, enter the requested 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. • **HSM**: the integrated hardware security module (HSM). • **Local Trust Store**: the local trust store managed by PingFederate. Visible only when PingFederate is integrated with an HSM in hybrid mode. (For more information, see *Supported hardware security modules* on page 56 .)
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.)

   When finished, click **Next**.
3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To import a certificate and its private key

1. Click **Import**.

2. On the **Import Certificate** screen:
   a) Click **Choose file** to locate and select the applicable certificate.
   b) Enter the certificate password.
   c) Select the storage facility of the certificate from the **Cryptographic Provider** list.
      - **HSM**: the integrated hardware security module (HSM).
      - **Local Trust Store**: the local trust store managed by PingFederate.

      Applicable and visible only when PingFederate is integrated with an HSM in hybrid mode.
   d) Click **Next**.
3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To review a certificate

1. Select the applicable certificate by its serial number.
2. Review the selected certificate in the pop-up browser window.

   Note: If a certificate has been revoked, PingFederate indicates this problem in the certificate information window.

   When finished, close the pop-up window.
3. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To activate a certificate

1. Click **Activate** under **Action** for the desired certificate.

   These choices are enabled only if you have created or imported more than one certificate.

   To undo the activation, click **Cancel**.
2. Click **Save** to complete the process.

To create a certificate-authority signing request

1. Click **Certificate Signing** under **Action** for the desired 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.

   Tip: 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 CA signing.
2. On the **Certificate Signing** screen, select the **Generate Certificate Signing Request (CSR)** option and then click **Next**.
3. On the **Generate CSR** screen, click **Export** to save the CSR file on your system.

   When finished, click **Done**.
4. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To import a certificate authority response

1. Click **Certificate Signing** under **Action** for the relevant certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option and then click **Next**.
3. On the **Import CSR Response** screen, click **Choose file** to locate and select the CSR response file.

   When finished, click **Next**.
4. On the **Summary** screen, review the information and then click **Save** to complete process.

To export a certificate

1. Click **Export** under **Action** for the certificate you want to export.
2. On the **Export Certificate** screen, select the type of export and then click **Next**.
   - **Certificate Only**: an export of the certificate without its private key. This is default choice.
   - **Certificate and Private Key**: an export of the certificate with its private key.

     Caution: This export contains the private key of the certificate. You must also enter an encryption password.

   If the certificate is stored on an HSM, the **Certificate and Private Key** option does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file on your system.

   When finished, click **Done**.
4. Click **Cancel** to go back to the **Server Configuration** screen.

To delete a certificate

1. Click **Delete** under **Action** for the certificate you want to delete.

   This option does not appear if the certificate is in use. To enable deletion, add (if needed) and activate a different certificate for the administrative console and/or the runtime server. If the usage is not clear, click **Check Usage**.

   To undo the deletion, click **Undelete**.
2. Click **Save** to confirm the removal of the certificate.

Manage digital signing certificates and decryption keys

You can use PingFederate to create and maintain your server's signing certificates, which you may use to sign outgoing requests, responses, assertions, and access tokens. The same type of certificate is also used for decryption.

Caution: For best security, we recommend using separate certificates for signing and decryption.

After creating your certificates, if they are left to be self-signed certificates, you can optionally enable automatic certificate rotation.

To create a new certificate

1. Click **Create New**.
2. On the **Create Certificate** screen, enter the requested 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. • **HSM**: the integrated hardware security module (HSM). • **Local Trust Store**: the local trust store managed by PingFederate.

Field	Description
	Visible only when PingFederate is integrated with an HSM in hybrid mode. (For more information, see *Supported hardware security modules* on page 56 .)
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.)

When finished, click **Next**.

3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To import a certificate and its private key

1. Click **Import**.
2. On the **Import Certificate** screen:
   a) Click **Choose file** to locate and select the applicable certificate.
   b) Enter the certificate password.
   c) Select the storage facility of the certificate from the **Cryptographic Provider** list.
      - **HSM**: the integrated hardware security module (HSM).
      - **Local Trust Store**: the local trust store managed by PingFederate.

      Applicable and visible only when PingFederate is integrated with an HSM in hybrid mode.
   d) Click **Next**.
3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To review a certificate

1. Select the applicable certificate by its serial number.
2. Review the selected certificate in the pop-up browser window.

   Note: If a certificate has been revoked, PingFederate indicates this problem in the certificate information window.

   When finished, close the pop-up window.
3. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To activate a certificate

1. Click **Activate** under **Action** for the desired certificate.

   These choices are enabled only if you have created or imported more than one certificate.

   To undo the activation, click **Cancel**.
2. Click **Save** to complete the process.

To create a certificate-authority signing request

1. Click **Certificate Signing** under **Action** for the desired 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.

> **Tip:** 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 CA signing.

2. On the **Certificate Signing** screen, select the **Generate Certificate Signing Request (CSR)** option and then click **Next**.
3. On the **Generate CSR** screen, click **Export** to save the CSR file on your system.

   When finished, click **Done**.
4. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To import a certificate authority response

1. Click **Certificate Signing** under **Action** for the relevant certificate.
2. On the **Certificate Signing** screen, select the **Import CSR Response** option and then click **Next**.
3. On the **Import CSR Response** screen, click **Choose file** to locate and select the CSR response file.

   When finished, click **Next**.
4. On the **Summary** screen, review the information and then click **Save** to complete process.

To export a certificate

1. Click **Export** under **Action** for the certificate you want to export.
2. On the **Export Certificate** screen, select the type of export and then click **Next**.
   - **Certificate Only**: an export of the certificate without its private key. This is default choice.
   - **Certificate and Private Key**: an export of the certificate with its private key.

     > **Caution:** This export contains the private key of the certificate. You must also enter an encryption password.

   If the certificate is stored on an HSM, the **Certificate and Private Key** option does not apply.

   If the certificate was provided by your partner for signature verification or encryption, the **Certificate and Private Key** option also does not apply.
3. On the **Export & Summary** screen, click **Export** to save the certificate file on your system.

   When finished, click **Done**.
4. Click **Cancel** to go back to the **Server Configuration** screen.

To delete a certificate

1. Click **Delete** under **Action** for the certificate you want to delete.

   This option does not appear if the certificate is in use. To enable deletion, add (if needed) and activate a different certificate for the administrative console and/or the runtime server. If the usage is not clear, click **Check Usage**.

   To undo the deletion, click **Undelete**.
2. Click **Save** to confirm the removal of the certificate.

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.

Although optional, it is recommended that you turn on email notifications for certificate events in the **Server Configuration** > **Server Settings** > **Runtime Notification** 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.

Email notifications

Although optional, it is recommended that you turn on email notifications for certificate events in the **Server Configuration** > **Server Settings** > **Runtime Notification** 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 send email 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 by 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 **Server Configuration** > **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 **Server Configuration** > **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.

Managed SP connection to PingOne and signing certificate

PingFederate automatically rotates the signing certificate used by the managed SP connection.

Note: A managed SP connection to PingOne is a connection created by the **Initial Setup** wizard or the **Server Configuration** > **Connect to PingOne** 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

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 **Server Configuration** > **Cluster Management** screen. When the process completes, the administrative console prompts the administrators to decide whether to update PingOne or to disconnect from PingOne in a banner message.

If the console node is shutdown, it will create a new signing certificate provided that it is restarted during the rotation window.

Optionally, PingFederate can be configured to send reminders ahead of expiry on the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen.

Note: If the signing certificate should be manually rotated instead, disable automatic certificate rotation.

Manage certificate rotation settings

You manage certificate rotation settings for self-signed certificates in the **Server Configuration** > **Signing & Decryption Keys & Certificates** screen.

1. On the **Certificate Management** screen, click **Certificate Rotation** next to 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 this certificate (*the current certificate*).

   If you want to turn off certificate rotation for this certificate, clear the check box and click **Save**.
3. On the **Certificate Rotation** screen, modify the following fields as needed:

   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.

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.

Manage certificates from partners

You receive certificates from partners for the following purposes:

- Signature verification: validating the digital signature in the incoming messages from your partners.
- Encryption: encrypting the outgoing messages before delivering them to your partners.
- Back-channel authentication: authenticating inbound (SOAP) messages from your partners by their client certificates.

You manage certificates from partner within a connection as follows:

- To manage certificates for the purpose of signature verification:
  a) On the **Activation & Summary** screen, click **Signature Verification Certificate**.
  b) On the **Signature Verification Certificate** screen, click **Manage Certificates**.

  You can import, review, activate, export, or delete certificates on the **Manage Digital Verification Certificates** screen.
- To manage certificates for the purpose of encryption:
  a) On the **Activation & Summary** screen, click **Select XML Encryption Certificate**.
  b) On the **Select XML Encryption Certificate** screen, click **Manage Certificates**.

  You can import, review, activate, export, or delete certificates on the **Manage XML Encryption Certificates** screen.
- To manage certificates for the purpose of inbound (SOAP) back-channel authentication:
  a) On the **Activation & Summary** screen, click **SSL Verification Certificate**.
  b) On the **SSL Verification Certificate** screen, click **Manage Certificates**.

  You can import, review, activate, export, or delete certificates on the **Manage Verification Certificates** screen.

> **Tip:** 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 items.

To import a partner certificate

1. Click **Import**.
2. On the **Import Certificate** screen:
   a) Click **Choose file** to locate and select the applicable certificate from your partner.
   b) Select the storage facility of the certificate from the **Cryptographic Provider** list.
      - **HSM**: the integrated hardware security module (HSM).
      - **Local Trust Store**: the local trust store managed by PingFederate.

Applicable and visible only when PingFederate is integrated with an HSM from Thales in hybrid mode.

c) Click **Next**.

3. On the **Summary** screen, review the certificate information and then click **Save** to complete process.

To review a partner certificate

1. Select the applicable certificate by its serial number.
2. Review the selected certificate in the pop-up browser window.

   Note: If a certificate has been revoked, PingFederate indicates this problem in the certificate information window.

   When finished, close the pop-up window.
3. On the **Certificate Management** screen, click **Cancel** to go back to the **Server Configuration** screen.

To activate a partner certificate

1. Click **Activate** under **Action** for the desired certificate.

   These choices are enabled only if you have created or imported more than one certificate.

   To undo the activation, click **Cancel**.
2. Click **Save** to complete the process.

To export a partner certificate

1. Click **Export** under **Action** for the certificate you want to export.
2. On the **Export Certificate** screen, click **Next**.
3. On the **Export & Summary** screen, click **Export** to save the certificate file on your system.

   When finished, click **Done**.
4. Click **Cancel** to go back to the **Server Configuration** screen.

To delete a partner certificate

1. Click **Delete** under **Action** for the certificate you want to delete.

   This option does not appear if the certificate is in use. To enable deletion, add (if needed) and activate a different certificate for the administrative console and/or the runtime server. If the usage is not clear, click **Check Usage**.

   To undo the deletion, click **Undelete**.
2. Click **Save** to confirm the removal of the certificate.

Manage keys for OAuth and OpenID Connect

On the **Server Configuration** > **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 sign ID tokens, JWTs for client authentication, and OpenID Connect request objects. When static keys are enabled, complete the configuration as follows:

Digital signature algorithm	Active key	Previous key
P-256	Optional	Optional
P-384	Optional	Optional
P-521	Optional	Optional
RSA	Required	Optional

For detailed information about the algorithms, please refer to the *JSON Web Algorithms (JWA)* specification (tools.ietf.org/html/rfc7518).

1. On the **Server Configuration** > **Signing & Decryption Keys & Certificates** screen, create the required keys if they have not yet been created.

   Note: Certificates that have been configured with automatic certificate rotation are not eligible to be selected as static keys. You may disable automatic certificate rotation for them or create new keys. Additionally, when creating new RSA keys, ensure that the key size is 2,048 bits or larger.
2. On the **Server Configuration** > **OAuth & OpenID Connect Keys** screen, 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.)
3. For the RSA digital signature algorithm, select an active key and optionally a previous key.

   If the desired key is not found, restart from step 1.

   (There is no default selection.)

   The required active key and the optional previous keys are published at the `/pf/JWKS` endpoint.
4. Optional: For any of the EC (elliptic curve) digital signature algorithms, select an active key and optionally a previous key.

   If the desired key is not found, restart from step 1. Alternatively, complete the configuration, create the desired keys, and then update the configuration afterward.

   (There is no default selection.)

   The the active key and previous keys (if configured) are published at the `/pf/JWKS` endpoint.
5. Optional: For any algorithm that you have selected an active key (with or without a previous key), select the **Publish Certificate** check box to publish the certificates associated with the active key and the previous key (if configured) at the `/pf/JWKS` endpoint.

   Tip: For each applicable key, its associated chain of certificates is published as the x5c parameter value. For more information, see *JSON Web Key (JWK)* (tools.ietf.org/html/rfc7517).

   (The **Publish Certificate** check boxes are not selected by default.)
6. Click **Save**.

Important: When static keys are enabled, PingFederate uses only static 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. EC algorithms that have not been configured with an active static keys are hidden.

For existing clients and IdP connections, if you have previously selected a certain EC algorithm (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 key for such EC algorithm (**P-256** in this example), transactions that involves the no longer supported EC 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 certificate revocation

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 Manage Certificate Revocation 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 74.)

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.

On the **Server Configuration** > **Certificate Revocation Checking** screen, modify settings for CRL checking and OCSP, as needed.

Field Descriptions (For OSCP Checking)

Field/Selection	Description
Enable OCSP	Turns on OCSP certificate-revocation checking.
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 unchecked (the default), 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 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).
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.
Responder Timeout	The allowable response time before the OCSP Responder URL is considered unavailable and processing continues (see "OCSP Responder is Unavailable," below).
Certificate is Unknown	The certificate does not fall under the purview of the CA associated with the OCSP Responder. The drop-down choices indicate whether an unknown certificate is to be considered valid or not, or whether to try CRL checking.
OCSP Responder is Unavailable	Indicates what action to take if the Responder cannot be reached.

Field/Selection	**Description**
OCSP Responder Returns Error	Indicates what action to take if the Responder returns an error.
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.

Field Descriptions (For CRL Checking)

Field/Selection	**Description**
Enable CRL Checking	Enables CRL revocation checking (the default). **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.
Treat Unretrievable CRLs as Revoked	If checked, PingFederate immediately aborts the processing associated with the certificate. If unchecked, the server treats the certificate as valid but continues trying to retrieve the CRL.
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.
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.
Verify CRL Signature	When checked (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 192).
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

Starting with PingFederate 8.3, 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: Once 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.

1. Install and configure the HSM client and the existing PingFederate environment (see *Supported hardware security modules* on page 56)

> **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.

Once 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 **Server Configuration** > **SSL Server Certificates** screen.

   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 193.
3. Create new digital signing certificates and decryption keys on the **Server Configuration** > **Signing & Decryption Keys & Certificates** screen 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 198.
4. If your connections support outbound (SOAP) back-channel authentication by client certificates, create new SSL client certificates on the **Server Configuration** > **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 196.
5. If you are transitioning to a Thales HSM, export the trusted CA certificates from the local trust store and import them to your HSM on the **Server Configuration** > **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 192.
6. If you are transitioning to a Thales HSM, for connections using the unanchored trust model, export the partner certificates 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*.)

   > **Tip:** You receive certificates from partners for the purposes of signature verification, encryption, or back-channel authentication. Partner certificates are managed in two buckets on a per-connection basis. While certificates for the purposes of signature verification and encryption are treated as one set of keys and certificates, client certificates for the purpose of inbound (SOAP) back-channel authentication are managed as another set of keys and certificates.

   For configuration steps, see *Manage certificates from partners* on page 204.

Manage 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.

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.

Tip: These capabilities allow you to quickly create connections with InCommon participants, to update the connections automatically or manually as the InCommon participants update their metadata, and to do so securely knowing PingFederate only commits changes to your connections after the digital signatures of the signed metadata are validated. For more information about InCommon participants, please refer to *www.incommon.org/participants*.

Use the **Server Configuration** > **Metadata URLs** screen to add, update, review, or remove SAML metadata URLs provided by your partners.

Add a new metadata URL

1. On the **Metadata URLs** screen, click **Add New URL**.
2. On the **URL** screen, enter the name and the URL of the metadata.
3. On the **Certificate Summary** screen, review the certificate information.
    a) If the metadata is not digitally signed (unsigned), click **Verify** to confirm that the unsigned metadata is reachable at the time of the configuration.
    b) If the metadata is signed but the certificate is provided outside of the metadata, click **Import** to upload the verification certificate.
4. Review the configuration, and then click **Done** and **Save**.

Tip: When creating a new or updating an existing SAML connection, if you enter a new metadata URL, the metadata URL is automatically added to the system.

Update an existing metadata URL

1. On the **Metadata URLs** screen, click the name of the applicable metadata URL.
2. Access the **URL** screen to update the name, the URL, or both, and then click **Next**.
3. Access the **Certificate Summary** screen, and then 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.

    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. Review the configuration, and then click **Done** and **Save**.

Review usage of or remove an existing metadata URL

1. On the **Metadata URLs** screen, click **Check Usage** for the applicable in-use metadata URL.

    The pop-up window provides a list of connections associated with the selected metadata URL.
2. On the **Metadata URLs** screen, use the **Delete**/**Undelete** workflow to remove or cancel the removal request for the applicable metadata URL.

    Note: You can only delete a metadata URL if it is not associated with any SAML connection.
3. Click **Save**.

Authentication

The configuration wizards under **Server Configuration** > **Authentication** allow administrators to manage basic authentication to the PingFederate server, when needed, as well as authentication needs for secured protocol transactions.

Configure application authentication

When you use the SAML 2.0 Attribute Query profile as an SP, password security is required between the application requesting attributes and the SP PingFederate server. Basic authentication is also required for applications making calls to PingFederate's Connection Management Service and optional for the SSO Directory Service (see *Web service interfaces* on page 567).

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 47 and */sp/startAttributeQuery.ping* on page 535).

In addition, authentication is required to access PingFederate runtime data via JMX (see *Runtime monitoring using JMX* on page 157) or to make *SOAP* calls to the Connection Management Web Service. Authentication is optional for the SSO Directory Service (see *Web service interfaces* on page 567).

Note: To help ensure network security, access to all of these services is deactivated when PingFederate is first installed.

On the **Server Configuration** > **Application Authentication** screen, administrators with the **Admin** administrative role can activate and configure authentication for the following services:

- Attribute Query
- JMX
- SSO Directory Service

To activate and configure authentication for the Connection Management Service, the administrators must be granted all three administrative roles: **Admin**, **Crypto**, and **User Admin**.

To enable access to a service:

1. Click **Activate** for the Service under Action.
2. Where required, enter an Id, Shared Secret, and Confirm Shared Secret for the service.

   You and the application developer must agree to these values.

   This step is optional for the SSO Directory Service; the Service can be active without requiring authentication (the default setting).
3. Repeat the steps above for other Services, as needed.
4. Click **Save**.

To change an application ID or password:

- Replace the existing information in the necessary field(s) and click **Save**.

To block access to an active service:

- Click **Deactivate** for the Service under Action and then click **Save**.

  Tip: 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 may wish to remove the corresponding WAR file or files from the `<pf_install>/pingfederate/server/deploy` and `<pf_install>/pingfederate/server/deploy2` directories, respectively:

  ```
  pf-ws.war
  pf-mgmt-ws.war
  ```

Manage Password Credential Validator instances

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.

- 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: 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 **Server Configuration** > **Server Settings** > **System Options** screen.

Choose a Password Credential Validator

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.

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.

Configure a Password Credential Validator instance

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:

- *Configure the LDAP Username Password Credential Validator* on page 213
- *Configure the PingOne Directory Password Credential Validator* on page 216
- *Configure the RADIUS Username Password Credential Validator* on page 217
- *Configure the Simple Username Password Credential Validator* on page 218

Configure the LDAP Username Password Credential Validator

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 an LDAP 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 `<pf_install>/pingfederate/server/default/conf/language-packs/pingfederate-messages.properties` file.

As needed, they may be localized using the PingFederate localization framework for an international audience.

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

1. Optional: Override authentication error messages.

   Note: This option may be required for an LDAP 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 LDAP 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.

Click **Move up** and **Move down** 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 LDAP 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.
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 *Extend the contract for the credential validator* on page 218). • 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 345). • 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 *Apply policy contracts or identity profiles to authentication policies* on page 240).

Field	Description
	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® 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 the Composite Adapter* on page 519 or **Incoming User ID** in *Specify an incoming user ID* on page 235, 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.
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	The LDAP attribute containing the PingID username of the users.

Field	Description
(for password reset)	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.
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.

Configure the PingOne Directory Password Credential Validator

The PingOne® Directory Username Password Credential Validator verifies credentials stored in the PingOne directory.

Note: The PingOne Directory Password Credential Validator requires a PingOne Enterprise account. For more information, see *Manage PingOne Directory Users*.

On the **Instance Configuration** screen, configure per-instance settings that suit your use cases.

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 account, see *View or Renew Directory API Credentials* in the PingOne *Enterprise Administration Guide*.
Advanced Fields	
PingOne URL	The PingOne directory API. The default value is `https://directory-api.pingone.com/api`.

Field	Description
Authenticate by Subject URL	The relative path for user authentication. The default value is `/directory/users/authenticate?by=subject`.
Reset Password URL	The relative path for password reset. The default value is `/directory/users/password-reset`.
SCIM User URL	The relative path for searching users requesting password reset. The default value is `/directory/user`.
Connection Pool Size	The maximum size of the connection pool to PingOne directory. The default value is `100`.

Configure the RADIUS Username Password Credential Validator

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.

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**.
Shared Secret	The password shared between PingFederate and the RADIUS server used to encrypt passwords. 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 `<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.

Click **Move up** and **Move down** 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 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`.

Configure the Simple Username Password Credential Validator

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.

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.

Click **Move up** and **Move down** 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.

Extend the contract for the credential validator

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 the LDAP server, the PingOne® directory, or the 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.

- 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.

Finish the Password Credential Validator instance configuration

- To complete and save the configuration, click **Done** on the **Summary** screen and then **Save** on the **Manage Credential Validator Instances** screen.

Configure Active Directory domains or Kerberos realms

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:

1. Configure the AD environment to integrate with PingFederate (see *Configure the Active Directory environment* on page 220).
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 219).

   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 *Configure the Active Directory environment* on page 220).

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.

Configure the Active Directory environment

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 219).

Note: You must have *Domain Administrator* permissions to make the required changes.

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: "`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.

Add a domain

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.

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.
Domain Controller/Key Distribution Center Host Names (optional)	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. 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.

Field	Description
	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:** For help resolving connectivity issues, select the **Debug Log Output** check box on the **Manage Domain/Realm Settings** 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.

Manage domain connectivity settings

Use the **Manage Domain/Realm Settings** screen to change default security and logging settings for all configured Active Directory domains and Kerberos realms.

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 acknowledgment that packets are properly received. **Note:** If you choose this option, ensure that you restart PingFederate after saving the configuration.
Debug Log Output	When selected, sends verbose messages to the PingFederate server log for all interactions with the domain controllers or the Key Distribution Centers (KDCs).
AD Domain Controller/ Key Distribution Center Timeout (secs)	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.

Configure 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.

1. Open the `com.pingidentity.common.security.AccountLockingService.xml` file.

   If you have a PingFederate clustered environment, open this file on the console node.

   - To review or modify the maximum number of failed attempts before a user is locked out for a time period, look for the MaxConsecutiveFailures element value.

     Note: The per-instance setting in the HTML Form Adapter and the Username Token Processor overrides this property.
   - To review or modify the amount of time (in minutes) that a user is locked out when the MaxConsecutiveFailures threshold is reached, look for the LockoutPeriod element value.
2. Save the change.
3. Restart PingFederate.
4. If you have a PingFederate clustered environment, click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen.

Note that settings stored on the `com.pingidentity.common.security.AccountLockingService.xml` file are applicable to all services that make use of account lockout protection.

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.

Manage authentication selector instances

You manage authentication selectors in the **Selectors** > **Manage Authentication Selector Instances** screen.

- 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.

Choose a selector type

1. Enter a name and an ID for this authentication selector instance.
2. Select the desired type of authentication selector from the list.

Configure an authentication selector instance

The configuration of an authentication selector instance varies depending on the authentication selectors deployed on your server. For authentication selectors bundled with PingFederate, refer to one of the following topics:

- *Configure the CIDR Authentication Selector* on page 223
- *Configure the Cluster Node Authentication Selector* on page 224
- *Configure the Connection Set Authentication Selector* on page 224
- *Configure the HTTP Header Authentication Selector* on page 224
- *Configure the HTTP Request Parameter Authentication Selector* on page 225
- *Configure the OAuth Client Set Authentication Selector* on page 226
- *Configure the OAuth Scope Authentication Selector* on page 227
- *Configure the Requested AuthN Context Authentication Selector* on page 227

- To complete the configuration:
  a) Click **Done** on the **Summary** screen.
  b) Click **Save** on the **Manage Authentication Selector Instances** screen.

Configure the CIDR Authentication Selector

The CIDR Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the IP address of an incoming 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 Adapters or between a Kerberos Adapter and an X.509 Adapter. For example, use this selector in one or more authentication policies to route internal requests to an instance of the Kerberos Adapter.

1. Click **Add a new row to 'Networks'**, enter a network range; then click **Update**.
2. Optional: Repeat the previous step to add more network ranges.

   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.

   Examples

   An 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`.

   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.

An IPv6 network range

Enter `2001:db8::/123` to cover 32 IPv6 addresses, ranging from `2001:db8::` through `2001:db8::1f`.

3. 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.
4. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

Configure the Cluster Node Authentication Selector

The Cluster Node Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on the PingFederate cluster node that is servicing the request in one or more authentication policies. 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.

1. On the **Authentication Selector** screen, click **Next** to proceed to the next screen.
2. On the **Selector Result Values** screen, enter a node index value based on your cluster configuration.

   This node index value should represent one of your engine nodes.
3. Repeat the previous step to add the node index values for the rest of the engine nodes.
4. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

When an instance of the Cluster Node Authentication Selector is deployed in an authentication policy, each specified value forms its authentication policy path.

Configure the Connection Set Authentication Selector

The Connection Set Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the target 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.

1. Click **Add a new row to 'Connections'**.
2. Select an SP connection from the list and click **Update**.

   At runtime, the selector compares the target SP connection in the SSO request to SP connections selected here. If a match is found, the selector returns a result value of `Yes`.
3. 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 entry. Use the **Delete** and **Undelete** workflow to remove an entry or cancel the removal request.
4. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

Configure 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. 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.

1. Click **Add a new row to 'Results'**.
2. Enter an expression for use when inspecting the HTTP header value of the target HTTP header under **Match Expression**; then click **Update**.

   Wildcard entries are allowed; for example, `*Chrome*`.
3. Optional: Repeat the previous step to add more expressions.

   For example, the following entries identify the most common browsers:

Browser	Expression
Chrome	`*Chrome*`
Firefox	`*Firefox*`
iPhone	`*iPhone*`
iPad	`*iPad*`
Internet Explorer	`*MSIE*` **Tip:** For information about Internet Explorer 11 (or higher), see *User-agent string changes* from Microsoft (msdn.microsoft.com/library/hh869301.aspx).
Safari	`*Safari*`

   Display order does not matter.

   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.
4. Enter the type of HTTP header you want the selector to inspect in the **Header Name** field; for example, `User-Agent`.

   This field is *not* case-sensitive.
5. 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.
6. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

At runtime, the selector compares the HTTP header to the specified expression (or expressions). If a match is found, the selector returns a result value of `Yes`.

Configure the HTTP Request Parameter Authentication Selector

The HTTP Request Parameter Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on query parameter values. 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.

1. On the **Authentication Selector** screen:
   a) Enter the request parameter name in the **HTTP Request Parameter Name** field.

      The HTTP request parameter name is case-sensitive.
   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.

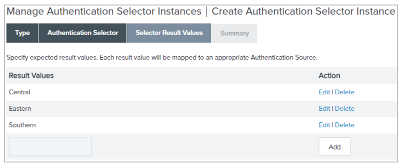
      The **Case-Sensitive Matching** check box is selected by default.
2. On the **Selector Result Values** screen, enter a request parameter value.

   Wildcard entries are allowed.

   Important: A more specific match is considered a better match and an exact match is considered the best match.
3. Optional: Repeat the previous step to add more request parameter values.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an entry. Click **Delete** to remove an entry.

   The following screen capture illustrates a configuration with three result values: Central, Eastern, and Southern.

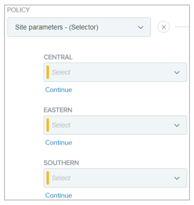
   
   Manage Authentication Selector Instances | Create Authentication Selector Instance
Type
Authentication Selector
Selector Result Values
Summary

   Specify expected result values. Each result value will be mapped to an appropriate Authentication Source.

Result Values	Action
Central	Edit I Delete
Eastern	Edit I Delete
Southern	Edit I Delete
	Add

4. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

When an instance of the HTTP Request Parameter Authentication Selector is deployed in an authentication policy, each specified value forms its own authentication policy path. Based on the previous sample configuration, the following screen capture illustrates the three policy paths when this selector instance is used in a policy.



Configure the 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.

Note: The OAuth Client Set Authentication Selector is only applicable to OAuth clients using the authorization code or implicit flow.

1. Click **Add a new row to 'Clients'**.
2. Select an OAuth client from the list and click **Update**.

   At runtime, the selector compares the client information in an OAuth request to the OAuth clients selected here. If a match is found, the selector returns a result value of `Yes`.
3. Optional: Repeat the previous step to add more OAuth clients.

Display order does not matter.

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.

4. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

Configure the OAuth Scope Authentication Selector

The OAuth Scope Authentication Selector enables PingFederate to choose configured authentication sources or other selectors based on a match found between the scopes of an OAuth authorization request and scopes configured in the PingFederate OAuth authorization server (AS).

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.

1. On the **Authentication Selector** screen, select the required scopes, scope groups, or both.

   At runtime, the OAuth Scope Authentication Selector compares the requested OAuth scopes to the scopes selected here. All of the selected scopes must match for the selector to return a result value of `Yes`.

   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.
2. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Authentication Selector Instances** screen.

Configure the Requested AuthN Context Authentication Selector

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).

1. On the **Authentication Selector** screen, 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
2. 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.

Note: The selector returns true when the specified value is an exact (case-sensitive) match to the authentication context (or one of the authentication contexts) requested by the SP or the RP.

b) Optional: Add more values to differentiate criteria for authentication selection.

Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an entry. Click **Delete** to remove an entry.

3. To complete the configuration:

a) Click **Done** on the **Summary** screen.

b) Click **Save** on the **Manage Authentication Selector Instances** screen.

When an instance of the Requested AuthN Context Authentication Selector is deployed in an authentication policy, each specified authentication context forms its authentication policy path.

Configure the Session Authentication Selector

The Session Authentication Selector enables PingFederate to choose a policy path at runtime based on whether the user already has a session for a particular source.

1. 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). Click **Manage Sessions** to configure authentication sessions.

2. Optional: Repeat the previous step to add more authentication sources.

Display order *could* matter.

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.

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.

3. 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.

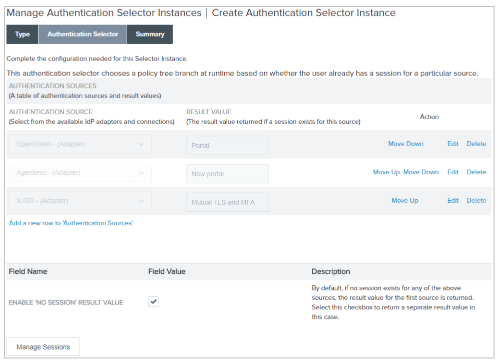
This check box is not selected by default.

4. To complete the configuration:

a) Click **Done** on the **Summary** screen.

b) Click **Save** on the **Manage Authentication Selector Instances** screen.

The following screen capture illustrates a configuration where three authentication sources are defined and the **Enable 'No Session' Result Value** check box is selected.


Manage Authentication Selector Instances | Create Authentication Selector Instance

Type | Authentication Selector | Summary

Complete the configuration needed for this Selector Instance.

This authentication selector chooses a policy tree branch at runtime based on whether the user already has a session for a particular source.

AUTHENTICATION SOURCES
(A table of authentication sources and result values)

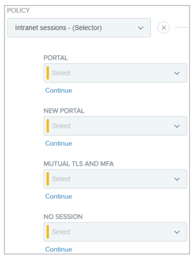
AUTHENTICATION SOURCE (Select from the available IdP adapters and connections)	RESULT VALUE (The result value returned if a session exists for this source)	Action
OpenToken - (Adapter)	Portal	Move Down Edit Delete
Agentless - (Adapter)	New portal	Move Up Move Down Edit Delete
X.509 - (Adapter)	Mutual TLS and MFA	Move Up Edit Delete

Add a new row to 'Authentication Sources'

Field Name	Field Value	Description
ENABLE 'NO SESSION' RESULT VALUE	✓	By default, if no session exists for any of the above sources, the result value for the first source is returned. Select this checkbox to return a separate result value in this case.

Manage Sessions

When this selector instance (named Intranet sessions) is placed in a policy, four policy paths are formed.


POLICY

Intranet sessions - (Selector)

PORTAL: Select — Continue

NEW PORTAL: Select — Continue

MUTUAL TLS AND MFA: Select — Continue

NO SESSION: Select — Continue

Configure a sample use case

The Session Authentication Selector enables PingFederate to choose a policy path at runtime based on whether the user already has a 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.

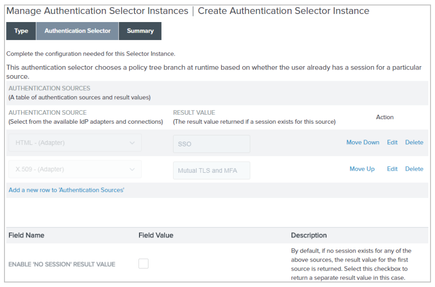
To fulfill this use case, follow these configuration steps:

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

Authentication source (adapter instance name)	Result value (policy path label)
X.509	Mutual TLS and MFA

The following screen capture illustrates the setup.


Manage Authentication Selector Instances | Create Authentication Selector Instance

Type | Authentication Selector | Summary

Complete the configuration needed for this Selector Instance.

This authentication selector chooses a policy tree branch at runtime based on whether the user already has a session for a particular source.

AUTHENTICATION SOURCES
(A table of authentication sources and result values)

AUTHENTICATION SOURCE (Select from the available IdP adapters and connections)	RESULT VALUE (The result value returned if a session exists for this source)	Action
HTML - (Adapter)	SSO	Move Down Edit Delete
X.509 - (Adapter)	Mutual TLS and MFA	Move Up Edit Delete

Add a new row to 'Authentication Sources'

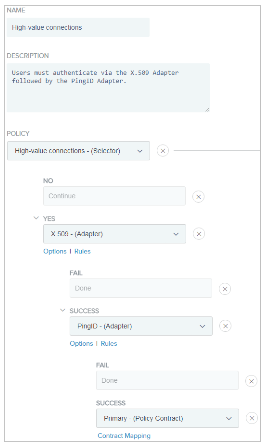
Field Name	Field Value	Description
ENABLE 'NO SESSION' RESULT VALUE	☐	By default, if no session exists for any of the above sources, the result value for the first source is returned. Select this checkbox to return a separate result value in this case.

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 *Specify an incoming user ID* on page 235.
   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.


NAME
High-value connections

DESCRIPTION
```
Users must authenticate via the X.509 Adapter
followed by the PingID Adapter.
```

POLICY
High-value connections - (Selector)

NO
Continue

YES
X.509 - (Adapter)
Options | Rules

FAIL
Done

SUCCESS
PingID - (Adapter)
Options | Rules

FAIL
Done

SUCCESS
Primary - (Policy Contract)
Contract Mapping

   l) Click **Done**.

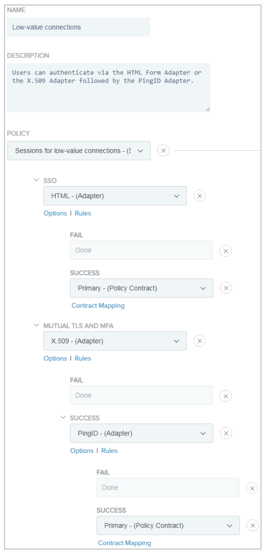
5. Define an authentication policy for low-value connections.

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 *Specify an incoming user ID* on page 235.

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.



NAME
Low-value connections

DESCRIPTION
```
Users can authenticate via the HTML Form Adapter or
the X.509 Adapter followed by the PingID Adapter.
```

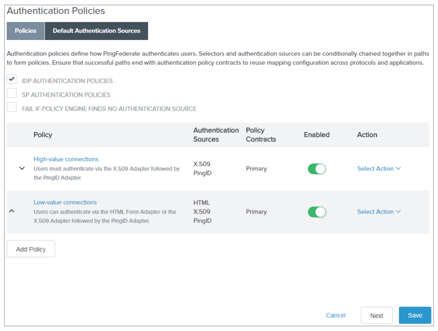
POLICY
Sessions for low-value connections - (S

SSO
HTML - (Adapter)
Options | Rules
FAIL
Done
SUCCESS
Primary - (Policy Contract)
Contract Mapping

MUTUAL TLS AND MFA
X.509 - (Adapter)
Options | Rules
FAIL
Done
SUCCESS
PingID - (Adapter)
Options | Rules
FAIL
Done
SUCCESS
Primary - (Policy Contract)
Contract Mapping

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.



Authentication Policies

Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

✓ IDP AUTHENTICATION POLICIES
☐ SP AUTHENTICATION POLICIES
☐ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
High-value connections Users must authenticate via the X.509 Adapter followed by the PingID Adapter.	X.509 PingID	Primary	●	Select Action
Low-value connections Users can authenticate via the HTML Form Adapter or the X.509 Adapter followed by the PingID Adapter.	HTML X.509 PingID	Primary	●	Select Action

6. 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**.)

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**). 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.

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.

Define authentication policies

You manage authentication policies and settings on the **Authentication Policies** screen.

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 **Server Configuration** > **Server 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 **Server Configuration** > **Server 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 4a*.
      - If the policy path is extended from an authentication source and it is the end of the path, select **Done**, which marks this 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**.)

- 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. Click **Next**.

8. 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.)

9. Click **Save**.

Specify an incoming user ID

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 or WS-Federation authentication request as the source. In this scenario, the incoming user ID is derived from the Subject element or the username parameter in the SAML 2.0 or WS-Federation authentication request, respectively.

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 or the username parameter in the SAML 2.0 or WS-Federation authentication request, respectively.

   If a request does not originate from a SAML 2.0 or WS-Federation authentication request or if the SAML 2.0 or WS-Federation authentication request does not include the optional Subject element or username 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.

Configure rules in authentication policies

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**.)

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.

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).

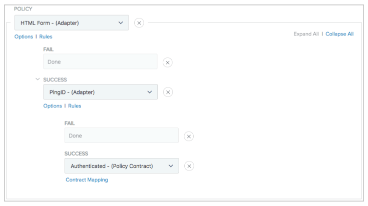
    For instance, if you have added two rules with labels **Contractors** (the first rule) and **Senior executives** (the second rule) to an authentication source, you should see the following results in the policy:

    - **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.

Define authentication policies based on group membership information

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:



POLICY
HTML Form - (Adapter)
FAIL
Done
SUCCESS
PingID - (Adapter)
FAIL
Done
SUCCESS
Authenticated - (Policy Contract)

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)

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.

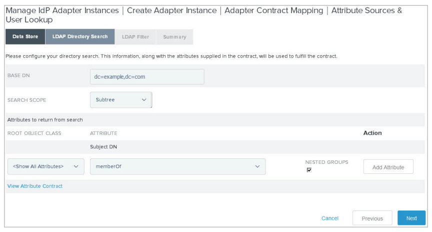
   Tip: Generally speaking, this step streamlines the process of deploying your authentication policies to additional groups of users later. In other words, when you are ready to roll out your authentication policies to more users, simply add the applicable groups as members of the new group. This way, 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.

      Manage IdP Adapter Instances | Create Adapter Instance | Adapter Contract Mapping | Attribute Sources & User Lookup

      Please configure your directory search. This information, along with the attributes supplied in the contract, will be used to fulfill the contract.

      BASE DN: dc=example,dc=com
      SEARCH SCOPE: Subtree

      Attributes to return from search

ROOT OBJECT CLASS	ATTRIBUTE	NESTED GROUPS	Action
	Subject DN		
<Show All Attributes>	memberOf	✓	Add Attribute

      View Attribute Contract

      (For more information, see *Define the IdP adapter contract* on page 325.)

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.

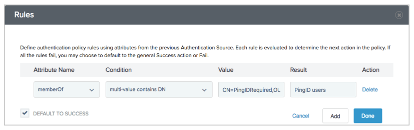
   c) Enter the DN of the new group created in *step 1* in the **Value** field; for example, `CN=PingIDRequired,OU=IT,DC=example,DC=com`.

   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:

   Rules

   Define authentication policy rules using attributes from the previous Authentication Source. Each rule is evaluated to determine the next action in the policy. If all the rules fail, you may choose to default to the general Success action or Fail.

Attribute Name	Condition	Value	Result	Action
memberOf	multi-value contains DN	CN=PingIDRequired,OU	PingID users	Delete

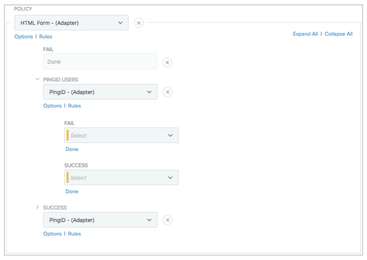
   ✓ DEFAULT TO SUCCESS

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 *Specify an incoming user ID* on page 235).

Your policy should be similar to the following sample:



```
POLICY
HTML Form - (Adapter)
    FAIL
    Done
    PINGID USERS
    PingID - (Adapter)
        FAIL
        Select
        SUCCESS
        Select
    SUCCESS
    PingID - (Adapter)
```

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.

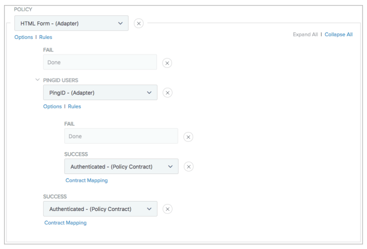
At runtime, PingFederate fulfills the authentication policy contract and carries on with the request.

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:



```
POLICY
HTML Form - (Adapter)
    FAIL
    Done
    PINGID USERS
    PingID - (Adapter)
        FAIL
        Done
        SUCCESS
        Authenticated - (Policy Contract)
    SUCCESS
    Authenticated - (Policy Contract)
```

9. Click **Done** to close the **Policy** screen.

10. On the **Authentication Policies** screen, click **Save**.

Note: Group membership is only one of the possible factors that you can use to define additional policy paths and their policy actions. Generally speaking, you can use any attributes available from the authentication source when configuring rules.

Apply policy contracts or identity profiles to authentication policies

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.

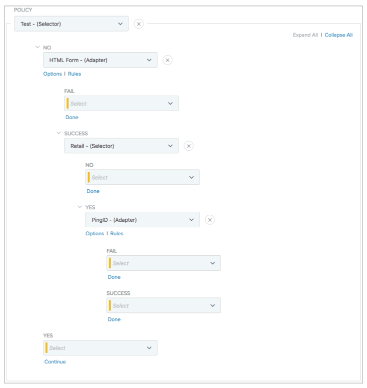
1. On the **Authentication Policies** screen, select the applicable authentication policy.
2. On the **Policy** screen, locate all closed-ended paths in the policy.

   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**.)

   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.

   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**

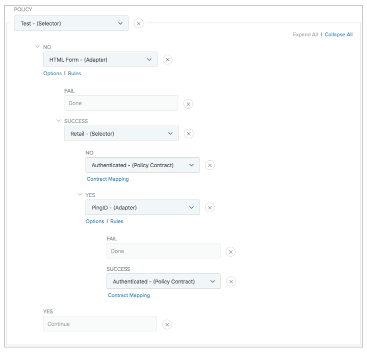
    Suppose your use case does not involve consumer authentication, registration, and profile management. It makes sense to select an authentication policy contract for the **PingID** > **Success** result, because the users have successfully met all your authentication requirements.

    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 *Configure contract mapping* on page 241).
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 *Configure local identity mapping* on page 242).
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:



POLICY
Test - (Selector)
Expand All | Collapse All
NO
HTML Form - (Adapter)
Options | Rules
FAIL
Done
SUCCESS
Retail - (Selector)
NO
Authenticated - (Policy Contract)
Contract Mapping
YES
PingID - (Adapter)
Options | Rules
FAIL
Done
SUCCESS
Authenticated - (Policy Contract)
Contract Mapping
YES
Continue

8. Click **Done** to close the **Policy** screen.
9. On the **Authentication Policies** screen, click **Save**.

Configure contract mapping

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 *Apply policy contracts or identity profiles to authentication policies* on page 240, 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.
3. Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract (see *Define issuance criteria for contract or local identity mapping* on page 242).
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.

Configure local identity mapping

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 397).
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, and attribute mapping expression (if enabled).
4. Optional: On the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract (see *Define issuance criteria for contract or local identity mapping* on page 242).
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.

Define issuance criteria for contract or local identity mapping

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.

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 Custom (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 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.

Map a policy contract to multiple use cases

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.

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.

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 293).

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` 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 sample use cases.

Configure an SP authentication policy for users from one IdP

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. In the **Service Provider** > **Server 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:

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:

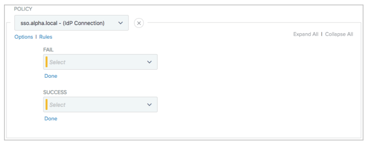
   - You map attributes to authentication policy contracts from authentication policies (*step 4* in this example).
   - You map attributes from authentication policy contracts to target applications through adapter mappings (*step 5* in this example).

   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:



POLICY

sso.alpha.local - (IdP Connection)

FAIL: Select

SUCCESS: Select

Each result forms its own policy path that requires further configuration.

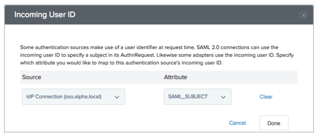
b) For the **sso.alpha.local** > **Fail** path, select **Done** as the policy action.

   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:



Incoming User ID

Some authentication sources make use of a user identifier at request time. SAML 2.0 connections can use the incoming user ID to specify a subject in its AuthnRequest. Likewise some adapters use the incoming user ID. Specify which attribute you would like to map to this authentication source's incoming user ID.

Source	Attribute	
IdP Connection (sso.alpha.local)	SAML_SUBJECT	Clear

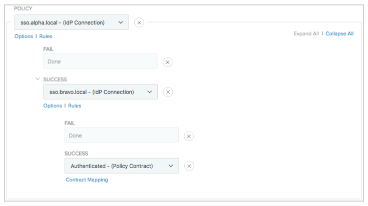
   Click **Done** to close the **Incoming User ID** dialog.

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:

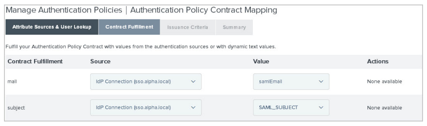


POLICY
sso.alpha.local - (IdP Connection)
FAIL
Done
SUCCESS
sso.bravo.local - (IdP Connection)
FAIL
Done
SUCCESS
Authenticated - (Policy Contract)

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.

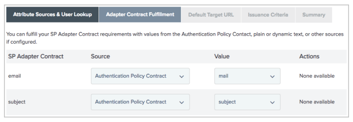
b) Select the applicable SP adapter instance (**Sample**) from the **Target Instance** list.

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:



You can fulfill your SP Adapter Contract requirements with values from the Authentication Policy Contact, plain or dynamic text, or other sources if configured.

SP Adapter Contract	Source	Value	Actions
email	Authentication Policy Contract	mail	None available
subject	Authentication Policy Contract	subject	None available

Tip: In this configuration, you are mapping attribute values from the authentication policy contract to the target application through the applicable SP adapter instance.

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

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

URL	Target Session
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 9.1.4, 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:

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.

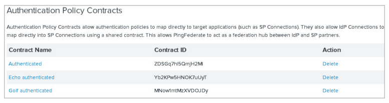
5. Create three authentication policy contracts, one for Alpha and Charlie, one for Echo, and one for Golf.

> **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	ZDSGq7hl5QmjH2MI	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.

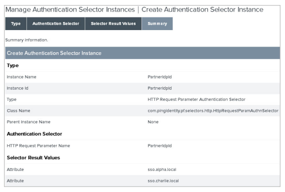
   Create an instance of the HTTP Request Parameter Authentication Selector.

   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

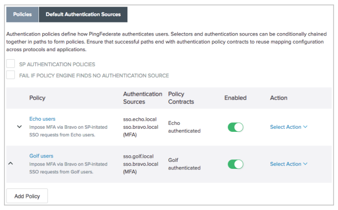
7. On the **Service Provider** > **Policies** screen, click **Add Policy** to define a policy to enforce the third-party authentication requirement for users from Echo (and then for users from Golf).

   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 244.

   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:



Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☐ SP AUTHENTICATION POLICIES

☐ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Echo users Impose MFA via Bravo on SP-initated SSO requests from Echo users.	sso.echo.local sso.bravo.local (MFA)	Echo authenticated	✓	Select Action
Golf users Impose MFA via Bravo on SP-initated SSO requests from Golf users.	sso.golf.local sso.bravo.local (MFA)	Golf authenticated	✓	Select Action

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*.

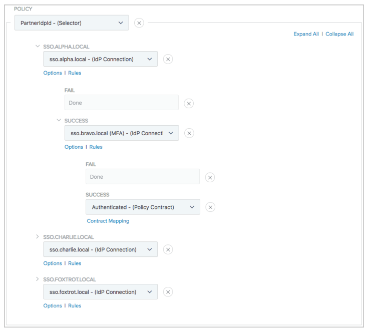
a) On the **Policy** screen, enter a name and a description for this policy, and select the instance of the HTTP Request Parameter Authentication Selector as the first policy action from the list.

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.

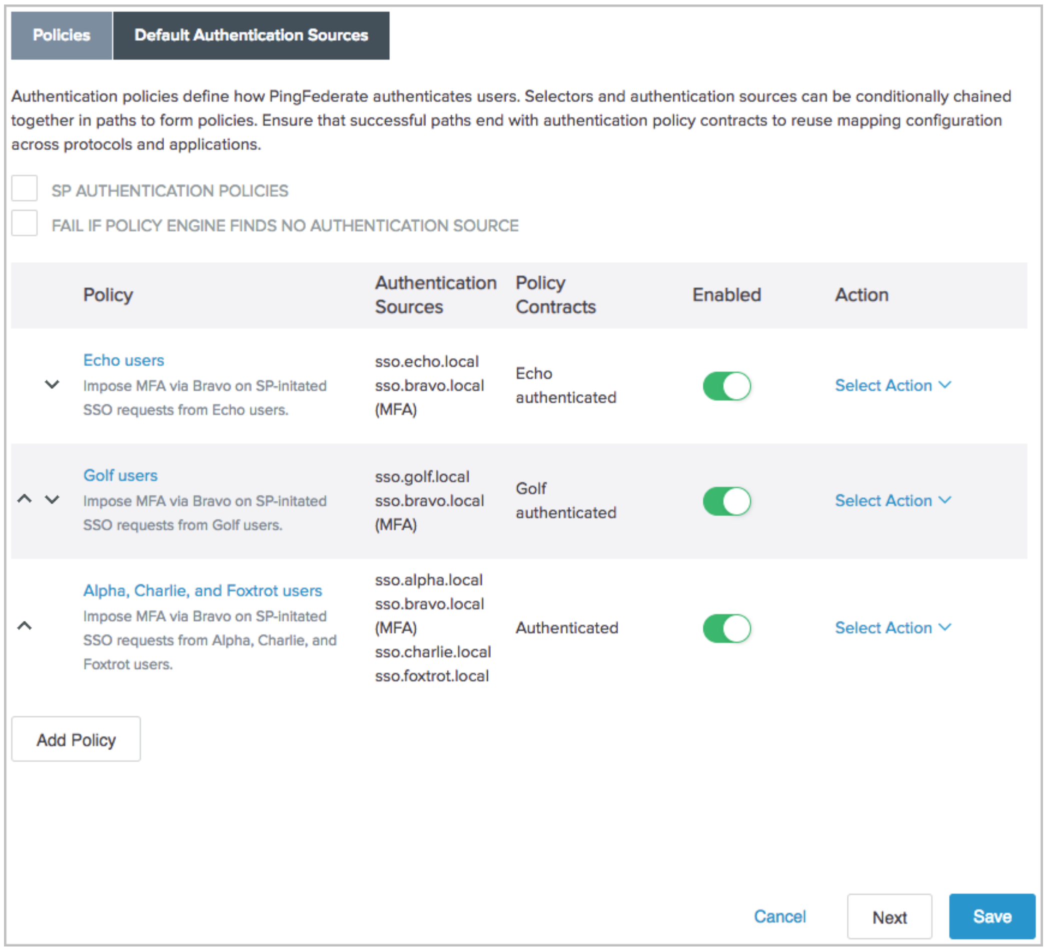
Your policy should be similar to the following sample:



POLICY
PartnerIdpId - (Selector)
SSO.ALPHA.LOCAL
sso.alpha.local - (IdP Connection)
FAIL
Done
SUCCESS
sso.bravo.local (MFA) - (IdP Connecti
FAIL
Done
SUCCESS
Authenticated - (Policy Contract)
SSO.CHARLIE.LOCAL
sso.charlie.local - (IdP Connection)
SSO.FOXTROT.LOCAL
sso.foxtrot.local - (IdP Connection)

(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:



Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

SP AUTHENTICATION POLICIES

FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Echo users Impose MFA via Bravo on SP-initated SSO requests from Echo users.	sso.echo.local sso.bravo.local (MFA)	Echo authenticated	●	Select Action
Golf users Impose MFA via Bravo on SP-initated SSO requests from Golf users.	sso.golf.local sso.bravo.local (MFA)	Golf authenticated	●	Select Action
Alpha, Charlie, and Foxtrot users Impose MFA via Bravo on SP-initated SSO requests from Alpha, Charlie, and Foxtrot users.	sso.alpha.local sso.bravo.local (MFA) sso.charlie.local sso.foxtrot.local	Authenticated	●	Select Action

e) Click **Save**.

9. Create adapter mappings.

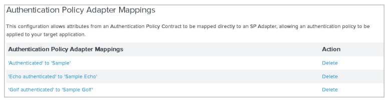
a) Go to the **Service Provider** > **Adapter Mappings** screen.

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 244.

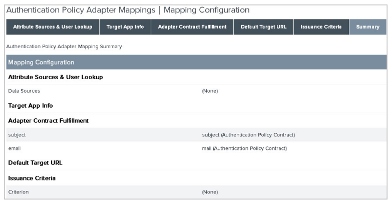
Your mappings should be similar to the following samples:


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

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)

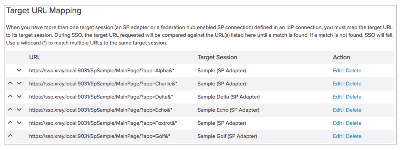
Tip: In essence, you are mapping attribute values from the authentication policy contracts to target applications through the applicable SP adapter instances.

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:



Target URL Mapping

When you have more than one target session (an SP adapter or a federation hub enabled SP connection) defined in an IdP connection, you must map the target URL to its target session. During SSO, the target URL requested will be compared against the URL(s) listed here until a match is found. If a match is not found, SSO will fail. Use a wildcard (*) to match multiple URLs to the same target session.

URL	Target Session	Action
https://sso.xray.local:9031/SpSample/MainPage/?app=Alpha&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Charlie&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Delta&*	Sample Delta (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Echo&*	Sample Echo (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Foxtrot&*	Sample (SP Adapter)	Edit | Delete
https://sso.xray.local:9031/SpSample/MainPage/?app=Golf&*	Sample Golf (SP Adapter)	Edit | Delete

11. Configure the following SSO URLs:

Partner	SSO URL
Alpha	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.alpha.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DAlpha%26t%3Da` The SSO URL has not changed. Based on the current configuration, because the target applications for Alpha and Charlie share the same SP adapter instance, the PartnerIdpId query parameter is required for the configured policy to route the request to the corresponding IdP connection.
Charlie	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.charlie.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DCharlie%26t%3Dc` The SSO URL has not changed. Based on the current configuration, because the target applications for Alpha and Charlie share the same SP adapter instance, the PartnerIdpId query parameter is required for the configured policy to route the request to the corresponding IdP connection.
Delta	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.delta.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%3Fapp%3DDelta%26t%3Dd` The SSO URL has not changed. **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).
Echo	`https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleEcho&TargetResource=https%3A%2F`

Partner	SSO URL
	`%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

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):

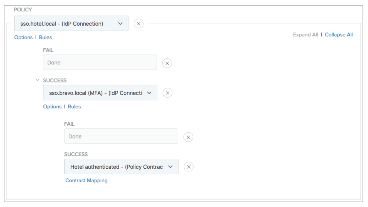
- An SP adapter instance:
  - Name: Sample Hotel
  - ID: sampleHotel
- A SAML 2.0 IdP connection:

- 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 9.1.4, 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:


POLICY
sso.hotel.local - (IdP Connection)
FAIL
Done
SUCCESS
sso.bravo.local (MFA) - (IdP Connecti
FAIL
Done
SUCCESS
Hotel authenticated - (Policy Contrac

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:

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:

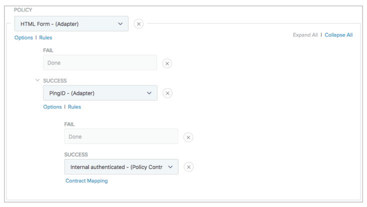
   - You map attributes to authentication policy contracts from authentication policies (*step 4* in this example).
   - You map attributes from authentication policy contracts to target applications through adapter mappings (*step 5* in this example).

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.

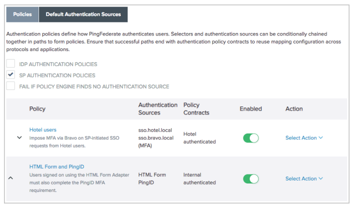
   In this example, the name and ID are both `Internal users`.

   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:



POLICY: HTML Form - (Adapter)
- FAIL: Done
- SUCCESS: PingID - (Adapter)
  - FAIL: Done
  - SUCCESS: Internal authenticated - (Policy Contr

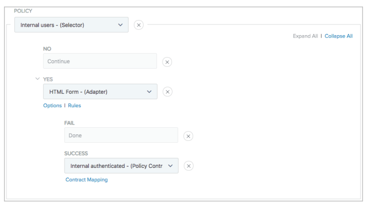


Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

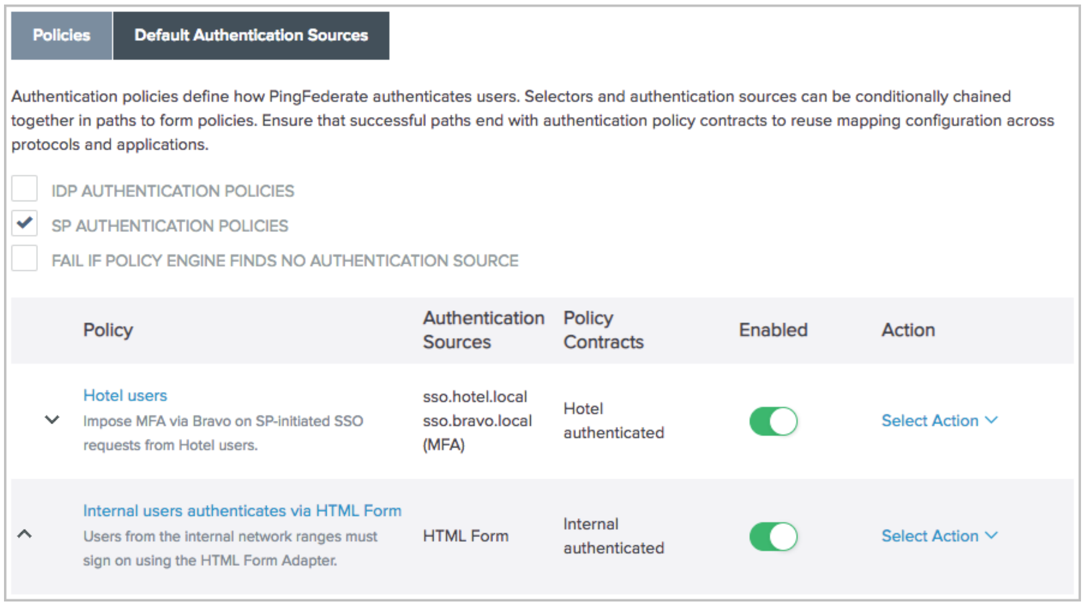
☐ IDP AUTHENTICATION POLICIES
✓ SP AUTHENTICATION POLICIES
☐ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Hotel users Impose MFA via Bravo on SP-initiated SSO requests from Hotel users.	sso.hotel.local sso.bravo.local (MFA)	Hotel authenticated	●	Select Action
HTML Form and PingID Users signed on using the HTML Form Adapter must also complete the PingID MFA requirement.	HTML Form PingID	Internal authenticated	●	Select Action

Figure 22: With PingID Adapter



POLICY: Internal users - (Selector)
- NO: Continue
- YES: HTML Form - (Adapter)
  - FAIL: Done
  - SUCCESS: Internal authenticated - (Policy Contr



Authentication policies define how PingFederate authenticates users. Selectors and authentication sources can be conditionally chained together in paths to form policies. Ensure that successful paths end with authentication policy contracts to reuse mapping configuration across protocols and applications.

☐ IDP AUTHENTICATION POLICIES

✓ SP AUTHENTICATION POLICIES

☐ FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Policy	Authentication Sources	Policy Contracts	Enabled	Action
Hotel users Impose MFA via Bravo on SP-initiated SSO requests from Hotel users.	sso.hotel.local sso.bravo.local (MFA)	Hotel authenticated	●	Select Action
Internal users authenticates via HTML Form Users from the internal network ranges must sign on using the HTML Form Adapter.	HTML Form	Internal authenticated	●	Select Action

Figure 23: 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

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 86).

Manage policy contracts

You manage authentication policy contracts (formerly known as connection mapping contracts) in the **Authentication Policies** > **Policy Contracts** screen.

- 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

You can enter or modify the contract name in the **Contract Info** screen. When complete, click **Done**.

Define contract attributes

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.

- 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

When you have finished configuring authentication policy contract, you can review the configuration on the **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save**.

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

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 603.
   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

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.

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 Custom (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 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 apply to PingFederate user facing endpoints, such as the profile management page and the grant management endpoints. This configuration is optional. Depending on the inactivity and maximum timeout values, previously authenticated users are redirected back to the authentication sources (IdP adapter instances or IdP connections) when the thresholds are reached.

Similarly, authentication sessions control when users are redirected back to the authentication sources on subsequent Browser SSO or OAuth authorization requests. This configuration is also optional. When configured, the global policy and timeout settings apply to all applications or all authentication sources, as described in the following table:

Authentication source	Use cases
IdP adapter instance	• IdP-initiated requests received at the `/idp/startSSO.ping` application endpoint • Authentication requests (AuthnRequest) received at the `/idp/SSO.saml2` protocol endpoint • OAuth authorization requests using IdP adapter instances for authentication (through the **IdP Adapter Mapping** configuration) • Adapter-to-adapter mappings (through the **Adapter-to-Adapter Mappings** configuration)
IdP connection	• SP-initiated requests received at the `/sp/startSSO.ping` application endpoint • OAuth authorization requests using IdP connections for authentication (through the OAuth Attribute Mapping configuration)

Administrators can enable authentication sessions for all authentication sources, with or without specifying overrides for selected authentication sources.

Alternatively, administrators can enable authentication sessions for a few selected authentication sources (and optionally with their own sets of overrides).

The override options include:

- Disable or enable authentication session; PingFederate always redirects the users back to the authentication system.
- Override the idle timeout, the maximum timeout, or both, in minutes.

  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 period, the session is extended by the idle timeout value (another 60 minutes by default). A 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. Note that 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.
- 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.

Administrators can also leverage the SLO application endpoints (`/idp/startSLO.ping` and `/sp/startSLO.ping`) to terminate adapter sessions or to add sessions to the session revocation list as users sign off (or to do both). When the **Track Adapter Sessions for Logout** check box is selected, if a user triggers a web application to send a logout request to one of the SLO application endpoints, the user is logged out locally (including adapter sessions established through **Adapter-to-Adapter Mappings** configuration). If the applicable connections support SLO (for instance an SLO-enabled SAML 2.0 connection or a WS-Federation connection), logout requests are sent to the partners based on the protocol specifications. When the **Track Revoked Sessions on Logout** check box is selected, PingFederate adds the associated sessions to the session revocation list as users sign off. This allows other systems, such as PingAccess®, to query the validity of a given session.

Configure application sessions

1. Go to the **Identity Provider (or Service Provider) Sessions** screen.
2. Optional: Configure the timeout settings under **Application Sessions**.
   a) Optional: Modify the default inactivity timeout value in the **Idle Timeout (Minute)** field.

      The default value is `60` (minutes).
   b) Optional: Modify the default maximum lifetime of an authentication session in the **Max Timeout (Minutes)** field.

      The default value is `480` (minutes, which is 8 hours).
3. Click **Save** to keep your configuration.

Configure authentication sessions

1. Go to the **Identity Provider (or Service Provider) Sessions** screen.
2. Optional: Select the **Track Adapter Sessions for Logout** check box to enable SLO for all adapter instances on a per-user basis.

   (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.

   Note: PingFederate always adds the sessions to the revocation list, even if an error occurs to the logout requests.

   This check box *is* selected by default for new installations.

Note: If your use cases involve OAuth requests, consider enabling the **Check session revocation status** option for 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 305).

4. Optional: Update the default value of the **Session Revocation Lifetime (Minutes)** field.

   This field defines the amount of time in minutes until the revoked sessions are removed from the revocation list for optimal performance.

   Important: The value should match or exceed the maximum session lifetime of the relying parties.

   The default value is `490` (minutes).
5. Optional: Configure the global policy and settings under **Authentication Sessions**.
   a) Select the **Enable Sessions for All Authentication Sources** check box to enable 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.

   b) Optional: Modify the default inactivity timeout value in the **Idle Timeout (Minute)** field.

      The default value is `60` (minutes).

   c) Optional: Modify the default maximum lifetime of an authentication session in the **Max Timeout (Minutes)** field.

      The default value is `480` (minutes, which is 8 hours).
6. 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 the 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 the 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.

   c) 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 then enter values into the **Idle Timeout** and **Max Timeout** fields, in minutes.

      The **Override Timeouts** check box is not selected by default. The **Idle Timeout** and **Max Timeout** settings have no default values.

   d) 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.

      The **Authentication Context Sensitive** check box is not selected by default.

   e) Click **Add**.

Repeat these steps to configure individual policy and settings for additional authentication sources, 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.

7. Click **Save** to keep your configuration.

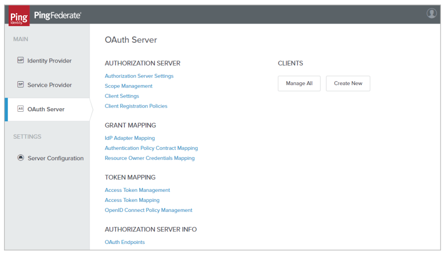
OAuth configuration

To use the OAuth authorization server (AS), start by enabling that role for PingFederate under **Server Configuration** > **Server 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

Use the **OAuth Server** menu to configure various settings to support the grant types that your applications require.



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 mappings between the authentication sources and the persistent grant contract.

   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 mappings between the persistent grant contract (or the authentication sources directly) and the attribute contract of your access token management instances using the **OAuth Server** > **Access Token Mapping** configuration wizard.

   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 69.)
6. For the SAML 2.0 or JWT profile for OAuth 2.0 authorization grants flow, configure a mapping on the **IdP Connection** > **OAuth Assertion Grant Attribute Mapping** screen.

   Note that this use case exchanges a SAML assertion or a JWT for an OAuth access token.
7. 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 **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.
8. Create one or more OAuth clients in the **OAuth Server** > **Client Management** screen.
9. Optional: Configure client settings and registration policies for dynamic client registration.
10. Optional: Configure client session management settings.

Enable the OAuth AS role

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*.

1. Go to the **Server Configuration** > **Server Settings** > **Roles and 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

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.

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 **Server Configuration** > **Server 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.

Field	Description
Persistent Grant Lifetime (Blank for Indefinite)	The amount of time in days (the default unit of time), hours, or minutes that PingFederate stores persistent grants. This field has no default value. Leave blank to maintain grants indefinitely. Enter an integer and select a unit of time to configure expiry information. **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.
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`.
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.
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.

Field	Description
	Note: You may override this setting in individual client records.
Allow Unidentified Clients to Request 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 value of the aud parameter (inside the JWT) against its base URL defined on the **Server Configuration** > **Server Settings** > **Federation Info** screen. If the values do not match, the authentication fails. Enter a separate base URL that PingFederate can take into consideration (in addition to the base URL defined on the **Federation Info** screen) when validating the aud parameter. Furthermore, if configured, the `/.well-known/openid-configuration` endpoint uses the **Token Endpoint Base URL** value as the base URL for the token endpoint. This field has no default value.
Persistent Grant Extended 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.

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.
Cross-Origin Resource Sharing Settings	Enter one or more 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/JWKS` • `/.well-known/openid-configuration` 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.

Define scopes

Where OAuth provides a mechanism to constrain the privileges associated with an access token, scopes provide a way to more specifically define the privileges requested and granted. Generally, a client specifies the scopes desired when asking for authorization. The issued access token is associated with the approved scopes.

For ease of management and subsequent client interactions, PingFederate provides the capability to create multiple groups of scopes. Essentially, a scope group is a *super scope* that a client can reference in applicable OAuth 2.0 protocol interactions. The scopes within a scope group can be downgraded (upon refresh) into application-specific scopes with fewer permissions.

Furthermore, PingFederate provides the flexibility to manage scopes and scope groups in two buckets, common scopes and exclusive scopes.

Common scopes

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) on a client-by-client basis in their client configuration.

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 using dynamic client registration. If a certain client requires a different set of common scopes (or scope groups), modify the client configuration using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created.

Exclusive scopes

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) on a client-by-client basis in their client configuration.

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 using dynamic client registration. If a certain client requires a different set of exclusive scopes (or scope groups), modify the client configuration using the administrative console, the administrative API, or the OAuth Client Management Service after the client has been created.

Note: The OpenID Provider configuration endpoint does not return exclusive scopes and scope groups.

Note: A scope (group) is either a common scope (group) or an exclusive scope (group). Duplicate scope (group) is 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.

Regardless of whether a scope (group) is created as a common scope (group) or an exclusive scope (group), a scope (group) represents access to a resource or API on the resource server (RS). Applicable scope (group) values require coordination with a developer or someone familiar with details of the RS OAuth implementation.

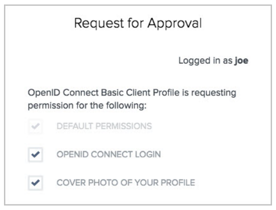
For clients support the OpenID Connect standard, add the following scopes for the purpose of requesting specific sets of claims from the OpenID Provider:

- openid
- address
- email
- phone
- profile

Tip: If most clients are allowed to use these scopes, create them as common scopes.

Scope Description

The **Scope Description** field controls the text appears when users are prompted for authorization; for example:



Request for Approval

Logged in as **joe**

OpenID Connect Basic Client Profile is requesting permission for the following:

- ✓ DEFAULT PERMISSIONS
- ✓ OPENID CONNECT LOGIN
- ✓ COVER PHOTO OF YOUR PROFILE

If you want to localize the display text, enter a unique alias in the **Scope Description** field; and then use the same alias in language resource files, located in the `<pf_install>/pingfederate/server/default/conf/language-packs` directory.

1. Go to the **OAuth Server** > **Scope Management** screen.
2. Optional: On the **Common Scopes** screen, configure any number of common scopes and scope groups for your clients.
   a) Enter a common scope and its description; then click **Add**.

      Repeat to define additional common scopes.
   b) Enter a common scope group and its description; then select one or more common scopes and click **Add**.

      Repeat to define additional common scope groups.

   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. Optional: On the **Exclusive Scopes** screen, configure any number of exclusive scopes and scope groups for your clients.
   a) Enter a exclusive scope and its description; then click **Add**.

      Repeat to define additional exclusive scopes.
   b) Enter a exclusive scope group and its description; then select one or more exclusive scopes and click **Add**.

      Repeat to define additional exclusive scope groups.

   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 common or exclusive scope value in production can cause runtime errors when a client request specifies the scope using the previously defined value. In addition, errors can occur when a requesting OAuth client is restricted to a scope whose value is no longer listed on this screen unless the affected client configuration is also updated.
4. Optional: On the **Default Scope** screen, enter a description for the default scope.

   The default scope is the permissions implied when no scope (group) values are indicated or in addition to any scope (group) values.
5. Click **Save** to retain all changes.

Related tasks
Localize messages for end users on page 147
Configure scope constraints on page 274

Related information
OpenID Connect, Requesting Claims using Scope Values (openid.net/specs/openid-connect-core-1_0.html#ScopeClaims)
OpenID Provider configuration endpoint on page 564

Configure client settings

Use the **Client Settings** configuration wizard to define various optional settings, such as metadata for OAuth clients and dynamic client registration settings.

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.

Define extended client metadata

On the **Extended Client Metadata** screen, add any number of client metadata fields to store additional information about OAuth clients, to support specific use cases, or both. This configuration is optional. If defined, the metadata fields become available to all OAuth clients in the **Client** configuration wizard.

Tip: When configuring individual clients, metadata values are optional and can be populated manually using the administrative console, the administrative API, or the OAuth Client Management Service.

If dynamic client registration is configured and enabled, developers can populate client metadata values by including them in the client registrations.

1. Go to the **OAuth Server** > **Client Settings** > **Extended Client Metadata** screen.
2. Add any number of client metadata fields.
   a) Enter the metadata name under **Metadata Parameter**.

      Once added and saved, the metadata name cannot be modified subsequently. You can however delete the metadata field completely.
   b) Optional: Enter a description for this metadata field under **Metadata Description**.
   c) If this metadata field should allow multiple values, select the check box under **Multivalued**.

      If you have initially added a single-valued metadata field, you can make it a multivalued metadata field by selecting the **Multivalued** check box later.

      If you have initially added a multivalued metadata field but you clear the **Multivalued** check box later, when you try to make changes to an OAuth client that has been configured with multiple values for this metadata field, the administrative console prompts you to reconfigure the client until only one value exists for this metadata field.
   d) Click **Add**.
   e) Optional: Repeat these steps to define additional metadata fields.

      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.

Configure dynamic client registration settings

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 272). If specific use cases require additional metadata, add them as extended client metadata on the **OAuth Server** > **Client Settings** > **Extended Client Metadata** 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 *Define an OAuth client data store* on page 183 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.

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.

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
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.
token_endpoint_auth_method	The client authentication method. Allowed values: • `none` • `client_secret_basic` • `client_secret_post` • `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` For more information, see *Mutual TLS Profiles for OAuth clients* (tools.ietf.org/html/draft-ietf-oauth-mtls-01).
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
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

Metadata field	Metadata description
	Note: RSASSA-PSS signing algorithms require an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. (For more information on HSM integration and static keys, see *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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.
grant_types	An array of one or more grant types, for which a client can request. Allowed values: • `authorization_code` • `password` (Resource Owner Password Credentials) • `refresh_token` (Use with `authorization_code` or `password`) • `client_credentials` • `implicit` • `urn:ietf:params:oauth:grant-type:jwt-bearer` (JWT Bearer Token) • `urn:ietf:params:oauth:grant-type:saml2-bearer` (SAML 2.0 Bearer Assertion)
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`

Metadata field	Metadata description	
	response type	**grant types**
	`code token`	`authorization_code` and `implicit`
	`id_token`	`implicit`
	`id_token token`	`implicit`
	`token`	`implicit`
client_name	A descriptive name for the client instance. This name appears when the user is prompted for authorization.	
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.	
jwks_uri, and	The URL of the JSON Web Key Set (JWKS) or the actual JWKS from the client.	
jwks	Only one of them is required if the client is configured to use the private_key_jwt client authentication method or to transmit request parameters in signed request objects (or to do both) 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 authorization requests and without the use of signed request objects for some other transactions. (For runtime processing, see *Authorization endpoint* on page 545.) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see *Supported hardware security modules* on page 56.)	

Configure scope constraints

On the **Scope Constraints** screen, optionally configure which scopes (or scope groups) developers can request when registering clients using dynamic client registration.

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.
   - If you select the **Restrict Common Scopes** check box without selecting at least one common scope, clients resulting from valid client registrations are configured without any common scopes.
   - If you select the **Restrict Common Scopes** check box with one or more applicable common scopes, developers must send client registrations with the desired common scopes. If they fail to do so, clients resulting from otherwise valid requests are also configured without any common scopes.
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.
- If you select one or more applicable exclusive scopes, developers must send client registrations with the desired exclusive scopes. If they fail to do so, clients resulting from otherwise valid requests are also configured without any exclusive scopes.

Restricting common scopes and allowing exclusive scopes are not mutually exclusive. You can configure both options based on your use cases. If you configure both options, developers must send client registrations with the desired common and exclusive scopes.

Note: The default scope is always allowed for and available to all clients.

It is worth noting that 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.

Manage client configuration defaults

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.

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 *Assertion Replay Prevention Service* on page 644).
Require Signed Request	Select this check box if the clients must submit authorization requests 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	Select a default Access Token Management (ATM) instance for this client.
Persistent Grants Expiration	Overrides the **Persistent Grant 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**

Field	Description
	• 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.
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 **Server Configuration** > **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`.
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 **Server Configuration** > **Server 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 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 *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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 *Manage keys for OAuth and OpenID Connect* on page 205. **Note:** While all settings on this screen can be overridden by client registration policies enforced during the registration, **ID Token Signing**

Field	Description
	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.

Select client registration policies

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.

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. Click **Move up** and **Move down** to adjust the execution order. Use the **Delete** and **Undelete** workflow to remove an existing instance or cancel the removal request.

Review client settings

When you have finished configuring client settings, you can review the configuration on the **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Save**.

Manage Client Registration Policy instances

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.

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**.

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

Use the **Client Registration Policy** configuration wizard to create a new or modify an existing Client Registration Policy instance.

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**.

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

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).

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**.

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

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.

- 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**.

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, an LDAP server, or a custom storage medium through the PingFederate SDK (see *Define an OAuth client data store* on page 183).

Configure an OAuth client

The **Client** screen provides controls over the usage and behavior of the applications requesting access to protected resources through the PingFederate OAuth AS.

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.
Client Authentication	The authentication method that the client uses. • Select **None** 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). • Select **Client Secret** 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. • Select **Client TLS Certificate** for mutual SSL/TLS 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 **Server Configuration** > **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 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 *Configure PingFederate properties* on page 98). • Select **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).

Field	Description
	• 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. **Note:** The underlying Assertion Replay Prevention Service is cluster-aware (see *Assertion Replay Prevention Service* on page 644).
Require Signed Request	Select the check box to require this client to send its authorization requests 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).
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 or to transmit request parameters in signed request objects (or to do both) 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 authorization requests and without the use of signed request objects for some other transactions. (For runtime processing, see *Authorization endpoint* on page 545.) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see *Supported hardware security modules* on page 56.)
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.

Field	Description
	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; the authorization consent page is not presented to the user 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.
Restrict Common Scopes	Restricts this client to access specific common scopes or scope groups. When selected, a list appears of all common scopes defined within the PingFederate AS. Select the common scopes or scope groups available for this client. **Note:** Selecting this box and not selecting any specific common scopes restricts this client's access to only the default scope defined in the **OAuth Server** > **Authorization Server Settings** screen and any explicitly allowed exclusive scopes and scope groups configured by the **Exclusive Scopes** setting. If a client requests a specific common scope or scope group (not selected in the list), an error is returned.
Exclusive Scopes	Allows this client to access specific exclusive scopes or scope groups. When selected, a list appears of all exclusive scopes defined within the PingFederate AS. Select the exclusive scopes or scope groups available for this client. **Note:** Selecting this box and not selecting any specific exclusive scopes allows this client's access to only the default scope defined in the **OAuth Server** > **Authorization Server Settings** screen and any common scopes and scope groups that have not been explicitly restricted by the **Restrict Common Scopes** setting. If a client requests a specific exclusive scope or scope group (not selected in the list), an error is returned.
Allowed Grant Types	Select at least one grant type that this client is allowed to use. Available grant types are: • **Authorization Code** • **Resource Owner Password Credentials** • **Refresh Token** • **Implicit** • **Client Credentials**

Field	Description
	• **Access Token Validation (Client is a Resource Server)** • **Extension Grants** 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	Select a 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.

Field	Description
Persistent Grants Expiration	Overrides the **Persistent Grant 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.
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 **Server Configuration** > **Server 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 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 *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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 *Manage keys for OAuth and OpenID Connect* on page 205. **Policy** Select a specific OpenID Connect policy from the list.

Field	Description
	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 for 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 305.) **Note:** If the **Track User Sessions for Logout** check box is selected in the **OAuth Server** > **Authorization Server Settings** screen, there are two additional fields: **PingAccess Logout Capable** and **Logout URIs**. **PingAccess Logout Capable** When selected, PingFederate sends (via the browser) logout requests to an OpenID Connect endpoint in PingAccess as part of the logout process (see *OpenID Connect endpoints* in the PingAccess documentation). This check box is not selected by default. **Logout URIs** Enter 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).

If you want to enable or disable the client, click the toggle switch.

2. Optional: On the **Extended Metadata** screen, add, remove, or edit one or more values for any metadata fields defined on the **OAuth Server** > **Client Settings** > **Extended Client Metadata** screen.

   Not applicable if no extended metadata is defined on the **OAuth Server** > **Client Settings** > **Extended Client Metadata** screen.
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 may 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. Persistent grants (and the associated attributes and their values, if any) remain valid until the grants expired or are explicitly revoked.

Manage IdP adapter grant mapping

Use the **OAuth Server** > **IdP Adapter Mapping** configuration to map values obtained from the authentication source into the persistent grantsThe 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.

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.

- 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.

Related concepts
Mapping OAuth attributes on page 69

Configure IdP adapter attribute sources and user lookup

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, click **Next**.

Define grant contract fulfillment for IdP adapter mapping

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grantsThe 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 Expiration** setting.

  - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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.

- LDAP/JDBC/Custom (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 LDAP, JDBC or Custom attributes identified for this 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 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.

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

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the IdP adapter mapping

When you have finished the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the **Manage Mappings** screen.

Configure IdP connection grant mapping

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.

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.

Related concepts
Mapping OAuth attributes on page 69

Choose an OAuth data store

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, select **No Data Store**; then click **Next**.

Configure contract fulfillment

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grantsThe 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 Expiration** setting.

  - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Custom**

  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 *Attribute mapping expressions* on page 593.

- **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.

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

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.

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 *Choose an identity mapping method* on page 417).
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.

Source	Description
	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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the OAuth attribute mapping summary

When you finish the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you finish, click **Done**.

Manage authentication policy contract grant mapping

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.

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.

1. On the **Authentication Policy Contract Mappings** screen, you may map from APCs into the persistent grants, modify a mapping as your OAuth use cases evolve, or remove a mapping when it is no longer applicable.

   - 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.
2. Optional: On the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to set up data store queries to fulfill the persistent grants.

   Follow the **Attribute Sources & User Lookup** configuration wizard to complete the setup. For configuration steps, see *Data store query configuration* on page 605.
3. On the **Contract Fulfillment** screen, configure fulfillment of each attribute in the persistent grant contract.

   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:

   - 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 Expiration** setting.

     - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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.

- Authentication Policy Contract

  When you make this selection, the associated **Value** list consists of the attributes (for example, subject) associated with the APC.
- LDAP/JDBC/Custom (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 LDAP, JDBC or Custom attributes identified for this 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.

4. On the **Issuance Criteria** screen, define criteria PingFederate can evaluate to determine whether to issue an access token for a user. This token authorization can be used to restrict who can access an OAuth-protected resource.

   For configuration steps, see *Define issuance criteria for policy contract mapping* on page 294.
5. On the **Summary** screen, click **Save** to complete the configuration.

Related concepts
Mapping OAuth attributes on page 69

Define issuance criteria for policy contract mapping

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Manage resource owner credentials grant mapping

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.

- 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.

Related concepts
Mapping OAuth attributes on page 69

Configure resource-owner attribute sources and user lookup

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, click **Next**.

Define resource-owner contract fulfillment

On the **Contract Fulfillment** screen, map values obtained from the authentication source into the persistent grantsThe 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:

- 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 Expiration** setting.

  - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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.
- 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.
- LDAP/JDBC/Custom (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 LDAP, JDBC or Custom attributes identified for this 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.

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

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the resource-owner credentials mapping

When you have finished the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the **Manage Mappings** screen.

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 may map from persistent grants (and optionally authentication sources, authentication policy contracts, authentication context, or Password Credential Validator instances to access tokens.

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 279).

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

Use the **OAuth Server** > **Access Token Management** configuration wizard to specify how the PingFederate OAuth AS manages access tokens.

- 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**.

Define an access token management instance

Define your access token management instance on the **Type** screen.

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

This configuration varies depending on the Type of the access token manager. The following sections provide information for token plug-ins bundled with PingFederate:

- *Configure reference-token management* on page 301
- *Configure JSON-token management* on page 302

Configure reference-token management

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 across all nodes despite any state server or subcluster setup.

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.

Field	Description
	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**.
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).

Related concepts
Adaptive clustering on page 634
Directed clustering on page 637

Configure JSON-token management

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.

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).

Field	Description
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 integrated with a hardware security module (HSM), additional RSASSA-PSS signing algorithms become available for selection. (For more information on HSM integration, see *Supported hardware security modules* on page 56.) Required if an asymmetric algorithm is selected in the **JWE Algorithm** list.
Active Symmetric Key ID	The ID of the symmetric key to use when producing JWTs using an HMAC-based algorithm. Required if an HMAC-based JWS algorithm is selected in the **JWS Algorithm** list.
Active Signing Certificate Key ID	The ID of the key pair and certificate to use when producing JWTs using an EC-based or RSA-based algorithm. Required if an EC-based or RSA-based JWS algorithm is selected in the **JWS Algorithm** list.
JSON Web Encryption (JWE) configuration	
JWE Algorithm	The algorithm used to encrypt or otherwise determine the value of the content encryption key; for example: • Symmetric algorithms (**Direct Encryption with symmetric key**, **AES ...Key Wrap**. and **AES-GCM ... key encryption**) • 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	

Field	Description
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.
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 566 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 566 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.)

Field	Description
	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.
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.

Manage session validation settings

When an OAuth client presents an access token for validation, PingFederate (the 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 (see *Introspection endpoint* on page 556)

If PingFederate is configured to manage authentication sessions, 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 (see *Back-Channel Session Revocation* on page 321).

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.

It is worth noting that, 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.

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.

   For information about authentication sessions, including configuration steps and settings, see *Configure authentication sessions* on page 262.

   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.

Feature	Description
	• 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. For more information, see *Client session management* on page 321.
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.

Define the access token attribute contract

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 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.

- To add a new entry, enter the desired value in the field and click **Add**.
- To modify an existing entry, use the **Edit**, **Update**, and **Cancel** workflow.
- To remove an existing entry, click **Delete**.

Manage resource URIs

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).

- 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.

Define access control

On the **Access Control** screen, you may restrict which OAuth clients are allowed to use this access token management instance.

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.

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

When you have finished the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Save**.

Manage access token mappings

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) 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.

- 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.

Related concepts
Mapping OAuth attributes on page 69

Configure access token attribute sources and user lookup

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, click **Next**.

Define access token attribute contract fulfillment

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:

- Adapter, Password Credential Validator, or IdP Connection

  If you have selected a specific IdP adapter instance, Password Credential Validator instance, or IdP connection 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**.
- 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 597.

  (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.)
- 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.
- LDAP, JDBC, or Custom (when a data store is used)

  Values are returned from your user-data store. When you make this selection, the **Value** list is populated by the LDAP, JDBC or custom attributes identified for this 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.

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

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.

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 Custom (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 *Manage access token mappings* on page 308).
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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the access token mapping

When you have finished the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the **Manage Mappings** screen.

Configure an OAuth assertion grant IdP connection

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).

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** to continue.

Define an attribute contract for the OAuth assertion grant

An attribute contract is a set of user attributes the IdP sends in the SAML assertion or JWTs for this connection. You identity these attributes on the **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.

To add an attribute:

1. Enter the attribute name in the text box.

   Attribute names are case-sensitive and must correspond to the attribute names expected by your partner.
2. Optional: Select the check box under Mask Values in Log.
3. Click **Add**.

To modify an attribute name or masking selection:

1. Click **Edit** under Action for the attribute.
2. Make the change and click **Update**.

   Note: If you change your mind, ensure that you click the Cancel link in the Actions column, 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.

Configure access token manager mappings

Use the **Access Token Manager Mapping** screen to associate one or more access token manager instances with this connection to define how access tokens are created.

To create a new access token manager mapping:

- Click **Create New Access Token Manager Mapping**.

To edit an existing access token manager mapping:

- Click on the access token manager instance and click the step you need to change.

To delete an access token manager mapping:

1. Click **Delete** next to the access token manager instance. (To undo the deletion, click **Undelete**.)
2. Click **Save** to confirm the deletion.

Select an access token manager instance

Use the **Access Token Manager** screen to select an access token manager instance to associate with this connection. Attributes are mapped from the connection's contract into the access token's contract to define the resulting content of created access token.

To select an access token manager instance:

- 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

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, select **No Data Store**; then click **Next**.

Define OAuth assertion grant contract fulfillment

On the **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**).
- **LDAP**, **JDBC**, or **Custom**

  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 *Attribute mapping expressions* on page 593.
- **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.

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

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.

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.
JDBC, LDAP, or Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review OAuth assertion grant attribute mapping configuration

When you finish the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you finish, click **Done**.

Review OAuth assertion grant configuration

When you finish the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you finish, click **Done**.

Configure OpenID Connect policies

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.

- 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

On the **Manage Policy** screen, enter the required information and configure optional settings for ID tokens issued under this policy.

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 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 566).
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

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.

- 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**.

Related concepts
UserInfo endpoint on page 566

Configure attribute scopes

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.

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**.

Related tasks
Define scopes on page 268

Related information
OpenID Connect specification, Requesting Claims using Scope Values (openid.net/specs/openid-connect-core-1_0.html#ScopeClaims)

Configure policy attribute sources and user lookup

You can optionally set up data store queries to supplement values returned from the source. This configuration is optional.

- 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 605.
- To skip this optional configuration, click **Next**.

Define policy contract fulfillment

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.)
- LDAP/JDBC/Custom (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 LDAP, JDBC or Custom attributes identified for this 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.

1. On the **OAuth Server** > **OpenID Connect Policy Management** screen, select the applicable policy.
2. On the **Contract Fulfillment** screen, choose a source and then choose (or enter) a value for each attribute in the contract.
3. Click **Next**.

Define issuance criteria for policy mapping

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.

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.
JDBC, LDAP, or Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the policy

When you have finished the configuration, you can review it on the **Summary** screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the **Manage Policies** screen.

> **Note:** If this is the first policy you are creating, you must designate it as the default before saving. You can change the default as needed when you create additional policies.

OAuth attribute mapping using a data store

This optional configuration is the same for all of the OAuth attribute-mapping task flows, including:

- *Configure OpenID Connect policies* on page 315
- *Manage resource owner credentials grant mapping* on page 296
- *Manage IdP adapter grant mapping* on page 285
- *Manage access token mappings* on page 308

A similar configuration is also used for attribute mapping in an IdP connection (see *Configure IdP connection grant mapping* on page 289) and when configuring an OAuth Assertion Grant connection (see *Configure an OAuth assertion grant IdP connection* on page 311).

(For additional information, see *Fulfillment by data store queries* on page 603.)

Choose a data store

On the **Data Store** screen, choose a data store for PingFederate to look up attributes.

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	• *Specify a JDBC database table and columns* on page 605 • *Enter a database search filter* on page 606
LDAP	• *Specify an LDAP directory and attributes* on page 607 • *Define encoding for LDAP binary attributes* on page 608 (optional) • *Enter an LDAP search filter* on page 608
Custom	• *Specify a custom source filter and fields* on page 609

Client session management

When an organization opens Web-based protected resources to their remote employees, business partners and customers, it has limited control over the end-user devices. To minimize security risk, both the IT administrators and end users desire session management with tight controls.

PingFederate provides an Asynchronous Front-Channel Logout endpoint and a Back-Channel Session Revocation Web Service to help OAuth clients, such as PingAccess®, to terminate sessions when end users log out and to prevent unauthorized access until the end users log in again.

PingAccess works out-of-the-box with PingFederate, taking full advantages of these two features.

Asynchronous Front-Channel Logout

Asynchronous Front-Channel Logout provides OAuth clients the capability to initiate single logout 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.

Configure Asynchronous Front-Channel Logout

1. Configure Asynchronous Front-Channel Logout settings.
   a) Select the **Track User Sessions for Logout** check box on the **OAuth Server** > **Authorization Server Settings** screen to enable Asynchronous Front-Channel Logout for OpenID Connect clients
   b) Optional: Select the **Track Revoked Sessions on Logout** check box on the **Identity Provider (or Service Provider)** > **Sessions** screen to add the PingFederate session to the revocation list.
2. Optional: Configure logout settings for each applicable OAuth client.
   a) Select the **PingAccess Logout Capable** check box to send (via the browser) logout requests to PingAccess.
   b) Enter additional logout endpoints at the relying parties under **Logout URIs** to send logout requests to close sessions on their end.

Related concepts
Sessions on page 261

Related tasks
Configure AS settings on page 265
Configure an OAuth client on page 279

Back-Channel Session Revocation

Back-Channel Session Revocation allows OAuth clients, such as PingAccess®, to query the revocation status of their sessions by sending HTTP GET requests to the session revocation endpoint on PingFederate at `/pf-ws/rest/sessionMgmt/revokedSris`.

To access the session revocation endpoint, a client must be granted access to the Session Revocation API. It must also authenticate with its client secret or client certificate and include in the request the session identifier, which can be obtained from the access token or the ID token.

Back-Channel Session Revocation also allows the clients to revoke sessions by sending HTTP POST requests to the same session revocation endpoint. This gives application developers the flexibility to revoke sessions based on the logic of their applications.

For each session added to the revocation list, PingFederate retains its revocation status for a configurable lifetime. Access control and authentication requirements to revoke sessions are identical to those to query for the revocation status.

Configure Back-Channel Session Revocation

1. For each applicable OAuth client, select the **Grant Access to Session Revocation API** check box in its client configuration screen.
2. For each OpenID Connect policy that has been applied to OAuth clients supporting session revocation, select the **Include Session Identifier in ID Token** check box.

   OpenID Connect policies are defined in the **OAuth Server** > **OpenID Connect Policy Management** screen.

   Alternatively, if authentication sessions are enabled, you may enable session validation for the applicable access token management instance such that the session identifier becomes available through the access tokens. OpenID Connect is not required in this case. For more information, see *Manage session validation settings* on page 305.
3. Optional: On the **Identity Provider (or Service Provider)** > **Sessions** screen, modify the **Session Revocation Lifetime** value.

   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.

Related concepts
Back-Channel Session Revocation Service on page 646
Sessions on page 261

Related tasks
Configure an OAuth client on page 279
Configure OpenID Connect policies on page 315
Configure authentication sessions on page 262

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 **Server Configuration** > **Server 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 **Server Configuration** > **Server 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 92).

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 86).

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 470.

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 72).

Under **Application 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. 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 **Application Integration** also include menu items for configuring plug-in **Token Processors** and optionally **STS Request Parameters** (see *IdP configuration for STS*).

Manage IdP adapters

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.

- 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**.

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 **Server Configuration** > **Server Settings** > **System Options** screen.

Create an IdP adapter instance

The first step in creating an adapter instance is choosing an adapter type.

- 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

Depending on the selected adapter, the **IdP Adapter** screen presents you with different configuration parameters.

- 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 498 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.

  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*).

  Applicable only if the adapter supports the notion of authentication context.

Invoke IdP adapter actions

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.

- Follow the on-screen instructions to complete the actions required.

Extend an IdP adapter contract

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.

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Set pseudonym and masking options

- 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

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,

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

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.

- 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**.

Fulfill an adapter contract

From the **Adapter Contract Fulfillment** screen, map values into the IdP adapter contract.

1. For a given attribute, select a source from the list.

   For more information about the **Source** list, refer to the following table:

Source	**Description**
Adapter	Values are returned from the IdP adapter without any customization.
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.
JDBC, LDAP, or Custom (if configured)	Values are returned from your attribute source. When you make this selection, the Value list is populated by the JDBC, LDAP, or Custom attributes you identified for this Attribute Source.

Source	Description
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. **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 **Server Configuration** > **Cluster Management** screen, and restart all nodes.
Text	The value is what you enter. You can enter text or 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 in the **Data Store** screen and *`attribute`* is any of the data store attributes you select. 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.

2. Specify a value.

   Not applicable when **Adapter** is the chosen from the **Source** list.
3. Repeat these steps until all attributes in the IdP adapter contract are mapped.

Define issuance criteria for adapter contract

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review an adapter contract

- On the **Summary** screen, review your configuration, modify as needed, and click **Done**.

Review and save an IdP adapter configuration

1. On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
2. 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 a default URL and error message

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 *Localize messages for end users* on page 147. 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 528). 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 132).

View IdP application endpoints

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate.

- 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 528 and *System-services endpoints* on page 542.

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 **Server Configuration** > **Server 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 154.

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.)
Metadata Service	`/` The default endpoint (empty path) from which partners can retrieve Auto-Connect metadata .
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 *Token translator mappings* on page 494).
	
	Note: If multiple token-processor instances of the same type are configured for the same connection or token-to-token mapping, a query parameter,

Service	URL and Description
	TokenProcessorId, must be added to either of these endpoints—see *Manage token processors* on page 472. (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 *Configure PingFederate properties* on page 98).

Virtual server ID support

For SAML connections using multiple virtual server IDs (see *Multiple virtual server IDs* on page 92), each virtual server ID has its own set of protocol endpoints. You may export a connection metadata for your partner on the **Server Configuration** > **Metadata Export** screen (see *Provide SAML metadata by file* on page 122).

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 542.)

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 *Federation planning checklist* on page 90).

If your agreement includes sending assertions containing attribute values from local data stores, you must define the required data stores (see *Manage data stores* on page 168).

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

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.

SP Connections

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's connection ID, and default virtual server ID; narrow by protocol and status; and perform the following actions.

Create a connection

Follow the **Connection** wizard to create a new connection to your SP partner.

Copy a connection

Follow the **Connection** wizard to create a new connection based on an existing (source) connection. This is most useful if the new connection and the source connection have many setting values in common. Note that the new connection stays disabled until you change its status.

Export a connection

Save the XML file as prompted. This is useful in situations where you want to make a backup of a connection prior to making changes to it.

Import a connection

Follow the **Import Connection** wizard to import a connection export. 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.

Export SAML metadata

Follow the **Export Metadata** wizard to generate a metadata for any SAML Browser SSO connection.

Update a SAML connection with metadata

Follow the **Import Metadata** wizard to update a SAML Browser SSO connection with metadata. 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.

Toggle the status of a connection

Click the toggle switch to enable or disable a connection.

Delete a connection

Delete a connection. You must click **Save** on the **SP Connections** screen to confirm your request. If you change your mind, click **Undelete** to cancel your request.

Override logging mode

Override the verbosity of runtime transaction logging for all SP connections.

Edit a connection

Open the connection by clicking on its name, select the setting that you want to reconfigure, and complete the change.

Resolve SP connection errors

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.

- 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

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.

1. Click **Import**.
2. On the **Import Connection** screen, browse to a connection XML file.
3. Optional: 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

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.

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	**1.** In the **Import Metadata** screen, select the **File** option. **2.** Choose the metadata file, and then click **Next**. **Note:** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata verification certificate on the **Import Certificate** screen, and then click **Next**. **3.** In the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **4.** Click **Next**.
A new metadata URL	**1.** On the **Import Metadata** screen, select the **URL** option. **2.** Enter a new metadata URL. **Note:** If you specify a URL that has already been entered into the system through the **Server Configuration** > **Metadata URLs** configuration wizard, the administrative console reminds you to select it from the **Existing URL Name** list. **3.** Click **Load Metadata**. **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**. **4.** In the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **5.** Click **Next**.
An existing metadata URL	**1.** On the **Import Metadata** screen, select the **URL** option. **2.** Select an existing metadata from the list, and then verify the URL in the **Selected Existing URL**. **3.** Click **Load Metadata**. **Note:** If there is a digital signature error, correct the issue using the **Server Configuration** > **Metadata URLs** configuration wizard. **4.** Click **Next**.

Note: If the endpoints in the metadata share the same base URL (protocol, hostname, and port), PingFederate 7.3 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

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. For information about using a template, refer to the *SaaS Quick Connection Guide*.

- 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

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.

- To configure a connection for secure browser-based SSO, select the **Browser SSO Profiles** check box.

  If you have selected multiple protocols on the **Server Configuration** > **Server 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 **Server Configuration** > **Server 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 **Server Configuration** > **Server 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

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.

- 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 *Configure standard IdP Discovery* on page 191).
- 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 47).

Import SP metadata

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
- Enter a new metadata URL
- Select an existing metadata URL

Tip: Using a metadata URL streamlines the configuration process. For example, by importing a metadata URL, you can quickly establish a Browser SSO connection to an InCommon-participating partner (see *www.incommon.org/participants*).

When you enter a new URL or select an existing metadata URL, PingFederate also enables the automatic update option and checks the metadata daily. If PingFederate detects changes in the partner's digital signing certificates, encryption key, or contact information, it updates the connection automatically. If you prefer to update the connection manually, you can clear the **Enable Automatic Reloading** check box.

The frequency is configurable through the **Reload Delay** field on the **Server Configuration** > **Server Settings** > **Metadata Lifetime** screen.

Although optional, it is recommended that you turn on email notifications for SAML metadata update events on the **Server Configuration** > **Server Settings** > **Runtime Notification** screen.

Note: If the metadata contains changes that require additional configuration, the email 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: These capabilities also lower the cost of maintaining connections with InCommon participants.

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	1. On the **Import Metadata** screen, select the **File** option. 2. Choose the metadata file, and then click **Next**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note:** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata verification certificate on the **Import Certificate** screen, and then click **Next**. 3. On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. 4. Click **Next**.
A new metadata URL	1. On the **Import Metadata** screen, select the **URL** option. 2. Enter a new metadata URL. **Note:** If you specify a URL that has already been entered into the system through the **Server Configuration** > **Metadata URLs** configuration wizard, the administrative console reminds you to select it from the **Existing URL Name** list. 3. Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. 4. Click **Load Metadata**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**.

Metadata medium	Steps
	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**. **5.** On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. **6.** Click **Next**.
An existing metadata URL	**1.** On the **Import Metadata** screen, select the **URL** option. **2.** Select an existing metadata from the list, and then verify the URL in the **Selected Existing URL**. **3.** Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. **4.** Click **Load Metadata**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note:** If there is a digital signature error, correct the issue using the **Server Configuration** > **Metadata URLs** configuration wizard. **5.** Click **Next**.

Identify the SP

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.

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 **Server Configuration** > **Server 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.

Field	**Description**
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.

Configure 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**.

IdP Browser SSO configuration steps

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 33). 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 SSO configuration steps

- *Choose SAML 2.0 profiles* on page 338
- *Set an SSO token lifetime* on page 338
- *Configure SSO token creation* on page 339
    - *Choose an identity mapping method* on page 339
    - *Set up an attribute contract* on page 340
    - *Manage authentication source mappings* on page 343
- *Configure protocol settings* on page 350
    - *Set Assertion Consumer Service URLs (SAML)* on page 350
    - *Specify SLO service URLs (SAML 2.0)* on page 352
    - *Choose allowable SAML bindings (SAML 2.0)* on page 353
    - *Set an artifact lifetime (SAML)* on page 353
    - *Specify artifact resolver locations (SAML 2.0)* on page 353
    - *Define signature policy (SAML)* on page 354
    - *Specify XML encryption policy (for SAML 2.0)* on page 433

WS-Federation SSO configuration steps

- *Set an SSO token lifetime* on page 338
- *Configure SSO token creation* on page 339
- *Define signature policy (SAML)* on page 354
    - *Choose an identity mapping method* on page 339
    - *Set up an attribute contract* on page 340
    - *Manage authentication source mappings* on page 343
- *Configure protocol settings* on page 350
    - *Define a service URL (WS-Federation)* on page 351

SAML 1.x SSO configuration steps

- *Set an SSO token lifetime* on page 338
- *Configure SSO token creation* on page 339
    - *Choose an identity mapping method* on page 339
    - *Set up an attribute contract* on page 340
    - *Manage authentication source mappings* on page 343
- *Configure protocol settings* on page 350
    - *Set Assertion Consumer Service URLs (SAML)* on page 350
    - *Set a default target URL (SAML 1.x)* on page 351
    - *Set an artifact lifetime (SAML)* on page 353
    - *Define signature policy (SAML)* on page 354

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 359). 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

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 38.)

- 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

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.

- 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.

Field	Description
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

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 *Federation hub use cases* on page 86 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.

- To continue, click **Configure ...**.

Choose an identity mapping method

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.

- For information about SAML selections, see *Select a SAML Name ID type* on page 339.
- For information about WS-Federation selections, see *Select a WS-Federation Name ID type* on page 340.

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

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 *User attributes* on page 79).

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.

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).

Option	Description
	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

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.

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

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 *Attribute contracts* on page 79). 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.

Protocol	Identity mapping	Attribute contract	SAML_SUBJECT	Additional attributes
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.
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.

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 79).

   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 attribute contract fulfillment* on page 345). 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 79).

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.

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

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 ACPs 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 222.

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 86.

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.

- 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**.

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

The first task of the mapping configuration is to map an adapter instance or an authentication policy contract to your connection.

- 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 77.

- 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.

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

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.

- 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 *Manage IdP adapters* on page 323).

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

When you multiplex one connection for multiple environments (see *Multiple virtual server IDs* on page 92), 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.

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.

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

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 325 or *Apply policy contracts or identity profiles to authentication policies* on page 240.

- 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 83 and *Attribute mapping with multiple data sources* on page 603.
- Select the **Use only ...** option if you do *not* require connection-level data store query.

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

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.

- 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 605.

  Repeat to add additional attribute sources.

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 attribute contract fulfillment

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 86).

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

- 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 *Bridging multiple IdPs to an SP* on page 88.

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 **Custom**

  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 *Attribute mapping expressions* on page 593.
- **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.

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.

Map default attribute contract fulfillment

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.

- 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 *Bridging multiple IdPs to an SP* on page 88.

  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 *Attribute mapping expressions* on page 593.
- **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.

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

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 *Multiple virtual server IDs* on page 92), 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 *Issuance criteria and multiple virtual server IDs* on page 596).

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.

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 Custom (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 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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the authentication source mapping

When you have finished adding a new or modifying an existing instance of an adapter or an authentication policy contract, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review the SSO token creation summary

When you are finished with the **Assertion Creation** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.

- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure protocol settings

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

- To continue, click **Configure Protocol Settings**.

Set Assertion Consumer Service URLs (SAML)

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.

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 593.

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 598 .

d) Click **Add**.

e) Optional: Repeat to add another message customization.

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)

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.

- Enter default destination in the **Default Target URL** field.

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)

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 189).

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.

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 593.

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 598 .

d) Click **Add**.
e) Optional: Repeat to add another message customization.

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)

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.

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.

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)

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.

- 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.

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)

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.

- Optional: Override the default value of the **Artifact Lifetime** field.

  The default value is 60 (seconds).

If you are editing an existing connection, you can update the artifact lifetime.

Specify artifact resolver locations (SAML 2.0)

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.

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.

If you are editing an existing connection, you may reconfigure any ARS endpoints.

Define signature policy (SAML)

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 *Digital signing policy coordination* on page 75).

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.

- To continue, select the option (or options) based on your partner agreement.

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)

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).

- To continue, select the option (and options) based on your partner agreement.

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

When you are finished with the **Protocol Settings** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.

- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review browser-based SSO settings

When you are finished with the **Browser SSO** primary task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure the Attribute Query profile

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 *Choose SP connection options* on page 334). 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 *Attribute Query and XASP*).

- To continue, click **Configure Attribute Query Profile**.

Define retrievable attributes

On this 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.

To add an attribute:

- Enter the attribute name in the text box and click **Add**.

To edit an attribute name:

1. Click **Edit** and make your change.
2. Click **Update**.

To delete an attribute:

- Click **Delete**.

Configure attribute lookup

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 portion of the attribute query configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

To configure an attribute source:

- Click **Add Attribute Source** and complete the setup steps (see *Choose a data store for attribute query* on page 356).

To modify an attribute source configuration:

1. Click the attribute source Description link.
2. Click **Save** on the screen you change.

   Note: Depending on what you change, you may need to modify dependent data in subsequent steps, as indicated. Click **Save** or **Done** when either of those options appears.

To reach this screen for editing:

1. Click **Identity Provider**
2. Click the connection name under SP Connections.

   Click**Manage All**, if needed, to see a full list of connections.
3. Click **Attribute Query** under the SP Connection tab.

4. Click **Configure Attribute Query Profile**.
5. Click **Attribute Source & User Lookup** on the Summary screen.

Choose a data store for attribute query

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.

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	• *Specify a JDBC database table and columns* on page 605 • *Enter a database search filter* on page 606
LDAP	• *Specify an LDAP directory and attributes* on page 607 • *Define encoding for LDAP binary attributes* on page 608 (optional) • *Enter an LDAP search filter* on page 608
Custom	• *Specify a custom source filter and fields* on page 609

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 attribute mapping fulfillment

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 *Bridging multiple IdPs to an SP* on page 88.

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/Custom

  Values are returned from your attribute source. When you make this selection, the Value list is populated by the LDAP, JDBC, or Custom attributes you identified for this Attribute Source.

  Note: PingFederate appends a description in parentheses for each data store lookup (see *Choose a data store for attribute query* on page 356).

- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 593). 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 356) 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

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Specify security policy

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 *Configure credentials* on page 359).

To configure attribute-query security policy for this partner:

- Check or clear the check boxes and click **Next** or **Done**.

Review the Attribute Query configuration

To reconfigure saved profiles:

1. Click the heading over the information you want to change.
2. Click **Done** on the screen containing your change.

   If you need to make additional changes, do so and continue by clicking **Done** until you reach the Attribute Query screen.
3. Click **Save** on the Attribute Query screen.

Configure credentials

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. Your connection configuration may involve any or all of the following:

- *Configure back-channel authentication (SAML)* on page 359
- *Configure digital signature settings* on page 362
- *Configure signature verification settings (SAML 2.0)* on page 362
- *Select an encryption certificate* on page 363
- *Select a decryption key (SAML 2.0)* on page 364

- To continue, click **Configure Credentials**.

Configure back-channel authentication (SAML)

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
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

Use case	Back-channel authentication requirements	Back-channel 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

Refer to the following topics for configuration steps:

- *Configure authentication requirements for outbound messages* on page 360
- *Configure authentication requirements for inbound messages* on page 360

Configure authentication requirements for outbound messages

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 *Manage SSL client keys and certificates* on page 196).

   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 for outbound authentication requirements:
   - If you need to make any changes, click the heading over the information you want to edit.
   - If you are creating a new connection and want to keep your configuration, click **Done**.
   - If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure authentication requirements for inbound messages

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 **Server Configuration** > **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 *Digital signing policy coordination* on page 75.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: 1. Enter the Subject DN of the certificate. 2. (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 *Manage certificates from partners* on page 204).

4. On the **Summary** screen for inbound authentication requirements:
   - If you need to make any changes, click the heading over the information you want to edit.
   - If you are creating a new connection and want to keep your configuration, click **Done**.
   - If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure digital signature settings

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.

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 198).

   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)

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.

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 **Server Configuration** > **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 75.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: 1. 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. 2. (Optional) Select the **Restrict Issuer** check box and enter the Issuer DN of the certificate. Alternatively, extract it from your partner's certificate. **Important:** Consider enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.
Unanchored	On the **Signature Verification Certificate** screen: 1. Select a primary 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 204). 2. Select a secondary certificate from the list. **Tip:** Use this field if your partner has sent you a new certificate to replace one that is ready to expire. The server will automatically verify against the secondary certificate when the primary one expires.

3. On the **Summary** screen for signature verification:

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Select an encryption certificate

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.

1. Optional: Change the default settings under **Block Encryption Algorithm**.

   Due to import control restrictions, the standard JRE distribution supports strong but not unlimited encryption. To use the strongest AES encryption, when permissible, download and install the appropriate version of "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" from the *Oracle Downloads* website (www.oracle.com/technetwork/java/javase/downloads/index.html).

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/)

Note: Under **Key Transport Algorithm**, **RSA-v1.5** is disabled for new connections for security reasons. If you are updating an existing connection that uses RSA-v1.5, consider changing the selection for increased security.

Note that the JCE provider libraries distributed with Gemalto SafeNet Luna SA HSM does not support the RSA-OAEP algorithm at the time of the PingFederate 7.1 R2 release. As a result, RSA-v1.5 is still allowed when deploying PingFederate with a Gemalto SafeNet Luna SA HSM.

2. 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 204).

Select a decryption key (SAML 2.0)

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.

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 *Manage digital signing certificates and decryption keys* on page 198).
2. Optional: Select the secondary XML decryption key from the list.

Review credential settings

When you finish configuring various settings under the **Credentials** task, you can review the configuration on its **Summary** screen.

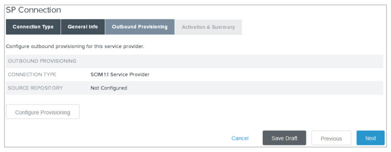
- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure outbound provisioning

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 85).

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 334).

- To continue, click **Configure Provisioning**.



Configure outbound provisioning for this service provider.

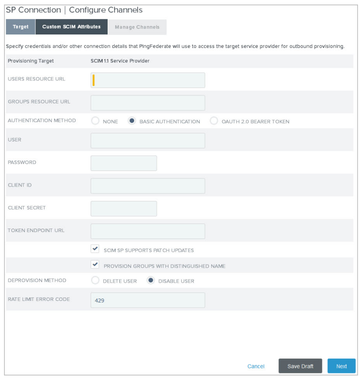
OUTBOUND PROVISIONING

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

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.



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®, 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.

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 Password	Valid credentials to access the partner's endpoint. Required if **Basic Authentication** is the selected authentication method.
Client ID, Client Secret, and Token Endpoint URL	Valid OAuth client credentials and token endpoint to access the partner's endpoint. Required if **OAuth 2.0 Bearer Token** is the selected authentication method.
SCIM SP Supports Patch Updates	Clear this 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.

Field	Description
	Some SCIM partners, including PingOne, 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.
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

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

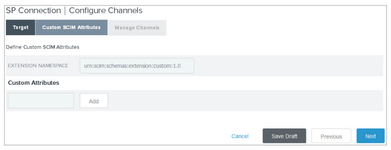
The following fragment illustrates a SCIM message supporting schema extension `urn:scim:schemas:extension:custom:1.0` with four attributes, one of each attribute type. The table afterward describes the details of each attribute.

```
{
  "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*.

1. Specify the URI of the schema extension in the **Extension Namespace** field.



SP Connection | Configure Channels

Target | Custom SCIM Attributes | Manage Channels

Define Custom SCIM Attributes

EXTENSION NAMESPACE: urn:scim:schemas:extension:custom:1.0

Custom Attributes

Add

Cancel | Save Draft | Previous | Next

   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.

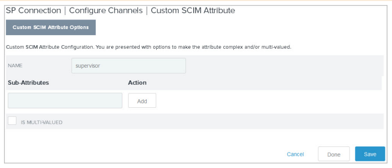
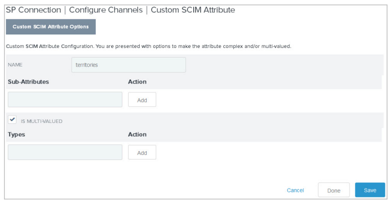
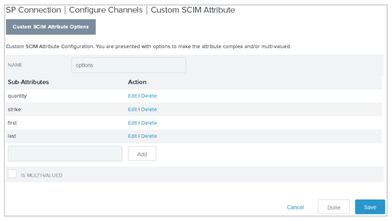
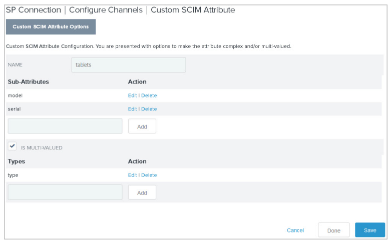
2. Enter an attribute name and click **Add** to add a custom attribute.

   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	1. Replace the current value in the **Name** field. 2. Click **Done**.

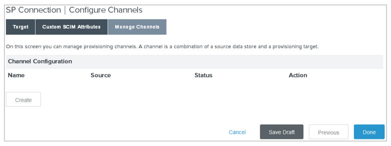
Task	Steps
	 SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute Options Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued. NAME: supervisor Sub-Attributes | Action: Add IS MULTI-VALUED Cancel Done Save
Set the attribute as a simple multivalued attribute	1. Select the **Is Multivalued** check box. 2. Click **Done**.  SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute Options Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued. NAME: territories Sub-Attributes | Action: Add ✓ IS MULTI-VALUED Types | Action: Add Cancel Done Save
Add sub-attributes to make the attribute a complex attribute	1. Enter a sub-attribute and click **Add**. (Repeat this step to add more sub-attributes as needed.) 2. Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the name of a sub-attribute. Use the **Delete** and **Undelete** workflow to remove a sub-attribute or cancel the removal request. 3. Click **Done**.  SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute 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 I Delete strike | Edit I Delete first | Edit I Delete last | Edit I Delete Add IS MULTI-VALUED Cancel Done Save
Add sub-attributes and set the attribute as a complex multivalued attribute	1. Enter a sub-attribute and click **Add**. (Repeat this step to add more sub-attributes as needed.) **Tip:** Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the name of a sub-attribute. Use the **Delete** and **Undelete** workflow to remove a sub-attribute or cancel the removal request. 2. Select the **Is Multivalued** check box. 3. Specify at least one value under the Types column for type, a reserved sub-attribute for a complex multivalued attribute. **Tip:** Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to the type value. Use the **Delete** and **Undelete** workflow to remove a type value or cancel the removal request. 4. Click **Done**.  SP Connection | Configure Channels | Custom SCIM Attribute Custom SCIM Attribute Options Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued. NAME: tablets Sub-Attributes | Action model | Edit I Delete serial | Edit I Delete Add ✓ IS MULTI-VALUED Types | Action type | Edit I Delete Add Cancel Done Save

When you finish editing custom attributes, click **Next** in the **Custom SCIM Attributes** screen.

Manage channels

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.



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

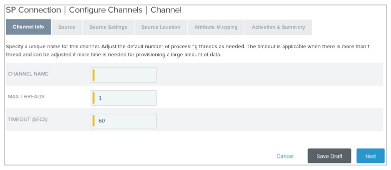
On the **Manage Channels** screen, you can perform the following tasks:

- 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

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.



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

1. Enter a channel name.

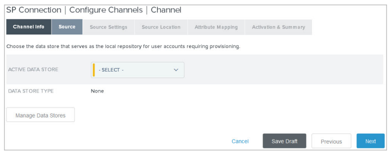
   If you are copying a channel, you must enter a new value in the **Channel Name** field.
2. Optional: Update the values for the **Max Threads** and **Timeout** fields.

   The **Timeout** value applies only when the **Max Threads** value allows multiple threads.
3. Click **Next**.

Identify the source data store

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.



Choose the data store that serves as the local repository for user accounts requiring provisioning.

ACTIVE DATA STORE	- SELECT -
DATA STORE TYPE	None

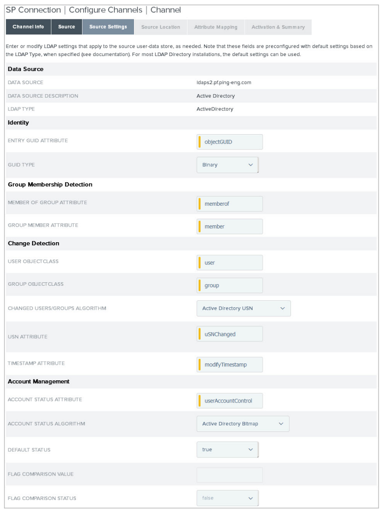
1. On the **Source** screen, choose the LDAP store to use for this channel.

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

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.



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.

Field	Description
	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.
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. 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**.

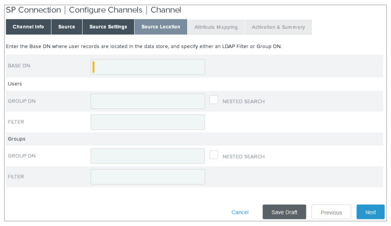
If you are using PingDirectory™, Microsoft Active Directory, and the Oracle Directory Server, in most cases no changes are needed on this screen unless your data store uses a customized schema.

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.)

1. Modify the settings, as needed.
2. Click **Next**.

Specify a source location

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.



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.

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:

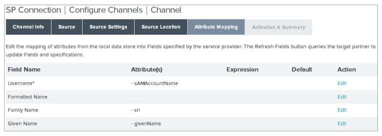
Object	Field description
Users	**Group DN** The distinguished name (DN) of a group in the user repository whose *member users* should be provisioned. (Optional) Select the **Nested Search** check box to include *users* that are members of the specified group through nested group membership. Nested group 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 369). **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.
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 369). **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.

Object	Field description
	Important: The **Group DN** field is ignored when a **Filter** field value is configured.

3. Click **Next**.

Map attributes

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.

For SaaS Connectors, refer to the connection-template setup steps in the *Quick Connection Guide* of the SaaS Connector for information about any special fields and how to map them.

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.

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.

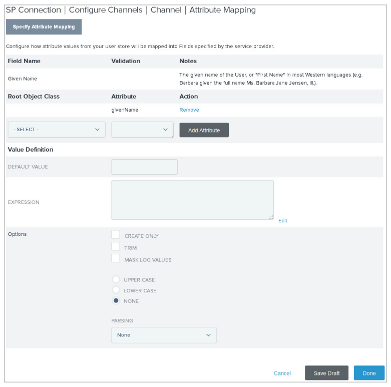
   The only exception is a field called LDAP Attributes Map, provided primarily to support PingOne$^{®}$-specific SCIM attributes. This field may contain multiple attributes without using OGNL.
4. Click **Next**.

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.

Define mapping information for a standard attribute


SP Connection | Configure Channels | Channel | Attribute Mapping

Specify Attribute Mapping

Configure how attribute values from your user store will be mapped into Fields specified by the service provider.

Field Name	Validation	Notes
Given Name		The given name of the User, or "First Name" in most Western languages (e.g. Barbara given the full name Ms. Barbara Jane Jensen, III.).

Root Object Class	Attribute	Action
	givenName	Remove
- SELECT -		Add Attribute

Value Definition

DEFAULT VALUE

EXPRESSION

Edit

Options: CREATE ONLY, TRIM, MASK LOG VALUES; UPPER CASE, LOWER CASE, [✓ NONE]

PARSING: None

Cancel Save Draft Done

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 steps above 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, or not.

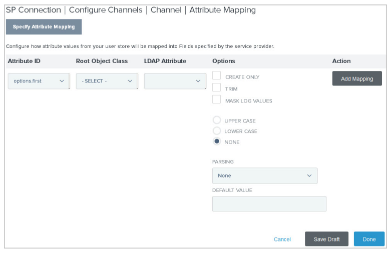
   Upper Case, **Lower Case**, or **None** – Transforms the attribute value(s) to the case indicated unless the **None** option is selected (the default).

   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.

   Tip: For SaaS Connectors not bundled with PingFederate, refer to your SaaS Connector *Quick Connection Guide* for instructions on mapping options or requirements for particular provisioning fields.
7. Click **Done**.

Define mapping information for a custom attribute:


SP Connection | Configure Channels | Channel | Attribute Mapping

Configure how attribute values from your user store will be mapped into Fields specified by the service provider.

Attribute ID	Root Object Class	LDAP Attribute	Options	Action
options.first	- SELECT -		☐ CREATE ONLY ☐ TRIM ☐ MASK LOG VALUES ○ UPPER CASE ○ LOWER CASE ● NONE PARSING: None DEFAULT VALUE:	Add Mapping

1. Select a sub attribute under **Attribute ID**.

   Note: Applicable only to *complex attributes* or *complex multivalued attributes* (see *Specify custom SCIM attributes* on page 366).

2. 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.

3. Select the source attribute from the class under **LDAP Attribute** and then click **Add Attribute**.
4. 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, or not.

   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.

   Tip: For SaaS Connectors not bundled with PingFederate, refer to your SaaS Connector *Quick Connection Guide* for instructions on mapping options or requirements for particular provisioning fields.

5. Optional: Enter a default value.
6. Click **Add Mapping**.

   Note: For complex attributes or complex multivalued attributes, repeat these steps to map additional sub attributes as needed.

7. Click **Done**.

Review channel settings

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).

- 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 *Configure outbound provisioning settings* on page 165). 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** in the **Activation & Summary** screen.

Review SP connection settings

When you finish creating or modifying a connection, you can review the connection settings and toggle the connection status on the **Activation & Summary** screen.

Important: When creating a new connection, the default connection status is **Inactive** when you reach the **Activation & Summary** screen.

Regardless of whether you choose to activate 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 (underneath the **Connection Status** field) is a sample URL that webmasters or web application developers at your site might use to invoke SSO for the connection (see *View IdP application endpoints* on page 328).

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, you must click **Save**.
- If you are modifying an existing connection (including toggling the connection status) and want to keep your changes, you must click **Save**.

Define 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 78).

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 343).

You can create or modify an SP affiliation under the **Identity Provider** menu (see *Manage SP affiliations* on page 376).

To create an SP affiliation:

1. Click **Identity Provider**.
2. Click **Create New** under SP Affiliations.

   Tip: See *Import affiliation metadata* on page 377 for subsequent steps.

To modify or delete an affiliation:

- See *Manage SP affiliations* on page 376.

Manage SP affiliations

You can manage SP affiliations on this screen.

To reach this screen for editing:

1. Click **Identity Provider**.
2. Click **Manage All** under SP Affiliations.

To begin creating a new affiliation:

- Click **Create Affiliation** (see the next sections for more information).

To delete an affiliation:

1. Click **Delete** under Action for the affiliation you want to delete.
2. Click **Save** to confirm the deletion (or click **undelete**).

To view or modify an affiliation:

- Click the affiliation ID.

Import affiliation metadata

An IdP may send a metadata file containing information that automatically specifies members of an SP affiliation for use in PingFederate.

- If you do not have a metadata file, click **Next**.

To import metadata:

1. Click **Browse** to locate and import the file and then click **Next**.
2. Review the information on the Create Affiliation page (see the next section).
3. Click **Save** on the Summary screen.

Enter affiliation information

Enter or modify basic information about an affiliation on the Affiliation General Info screen.

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.

Field descriptions

Field	Description
Affiliation ID	A unique identifier for this affiliation. This value serves as the Name ID qualifier for SAML assertions sent to affiliated SP partners.
Affiliation Owner	Any SAML 2.0 SP connection may serve as the Owner.

Manage affiliation membership

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.

- 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 343).
- 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

From the Affiliation Management Summary screen you can activate or deactivate an SP affiliation. You also save new affiliations on this screen, or you can click heading links to go back and modify information.

To change an Affiliation Status:

- Select either Active or Inactive and then click **Save**.

Important: Be sure to click **Save**. Otherwise, the status will not be changed.

To edit a connection:

1. Click the heading above the information you want to modify.
2. Make your change and click **Save**.

Configure SP Auto-Connect

When your SP partner is also using PingFederate 5 or higher (or is otherwise able to provide interoperable SAML 2.0 metadata via HTTP on demand), you may choose to use Auto-Connect™ for that partner (see *Using Auto-Connect*). This configuration can be shared among an unlimited number of SAML 2.0 partners.

Note: You enable the SAML 2.0 Auto-Connect profile under System Settings (see *Choose roles and protocols* on page 161).

Once Auto-Connect™ is enabled on your PingFederate server, complete the configuration from the Identity Provider menu. This configuration entails:

- Setting up a common connection for all Auto-Connect partners
- Establishing a list of SP partner domains authorized to use the connection

Initial setup for SP Auto-Connect

The basic configuration for Auto-Connect™ requires only:

- Defining a period of validity for assertions (assertion lifetime)
- Choosing a signing certificate for assertions and other SAML messages
- Configuring assertion-creation information

All other partner-connection specifications are handled automatically at runtime.

Specify an assertion lifetime for SP Auto-Connect

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.

Field Descriptions

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.

To change the default times:

- Optional: Edit the desired setting(s) and click **Next** or **Save**.

Choose a signing certificate for SP Auto-Connect

For Auto-Connect™ runtime processing, assertions and SLO messages must be signed, because they are sent over either the POST or redirect bindings (see *SAML 2.0 profiles* on page 38).

Note: The signing certificate is embedded in your server's Auto-Connect metadata (see *Auto-Connect* on page 94); there is no need to exchange certificates with your partners.

You can use the same certificate used for signing metadata (see *Configure metadata signing* on page 166). If you use a different certificate, ensure that it meets Auto-Connect validation requirements (see *Auto-Connect security model* on page 95).

To specify a certificate:

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 198).
2. 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 assertion creation for SP Auto-Connect

Configuring assertion creation for Auto-Connect™ is similar to configuring the same settings for regular partner connections.

- Click **Configure Assertion Creation** to continue.

  For configuration information, refer to sections under *Configure SSO token creation* on page 339.

Review SP Auto-Connect settings

When you finish configuring your SP Auto-Connect™ initial setup, you may choose to activate the common connection immediately on the Activation & Summary screen. (No runtime processing occurs until your partner's Auto-Connect gateway is also established and a user initiates an SSO or SLO event.)

Important: Regardless of whether you choose to activate a newly configured connection now or later, you must click **Save** on the Activation & Summary screen if you want to keep the configuration.

You can deactivate the connection at any time (for maintenance, for example). While a connection is inactive, all SSO or SLO transactions to or from Auto-Connect partners are disabled.

To change a Connection Status:

- Select Active or Inactive and then click **Save**.

To modify a setting:

1. Locate the currently configured setting under one of the summary headings and then click the subheading above the data.

   Note: Changes made to Auto-Connect settings will be out of sync, temporarily, with metadata caches that any currently active partners might be using. If your connection is in production, you might wish to lower your server's metadata lifetime in advance of making configuration changes (see *Configure metadata lifetime* on page 166).
2. Change the information and click **Save**, if available.

   If **Save** is not available, additional, dependent changes are required; click **Next** or **Done** until you reach a screen containing a **Save** button. Then click **Save** and continue as needed.

Specify allowed SP domains

This screen provides PingFederate with a list of trusted domain names of your Auto-Connect™ partners.

Normally, when PingFederate receives an authentication request from a domain in this list), the runtime engine completes the connection automatically using metadata obtained from a standard, public location— `http://saml.<domain_name>`. (See *Using Auto-Connect*.) Alternatively, if an Auto-Connect partner elects not to use the standard location, you can supply the applicable URL.

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

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 397).

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 **Server Configuration** > **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 **Server Configuration** > **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 385).

Manage local identities profiles

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.

- 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

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.

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

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.

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, or Twitter, 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.)

Define local identity fields

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.

- 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

On the **Field Configuration** screen, create a new or modify an existing local identity field.

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 *Configure local identity mapping* on page 242).

   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 verification options

Based on your customer IAM use cases, you can optionally offer users the opportunity to confirm the ownership of the email address associated with their accounts. This configuration is optional and can be configured on a per-local identity profile basis.

For registration, PingFederate sends a verification email message to the user as the user submits the registration request. The verification email 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 verification email message 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 sent to the applicable target applications through IdP authentication policies.

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 verification email 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.

   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. If you have not yet added information about your email server, click **Manage Email Server Settings**.

Configure registration options

Configure the registration experience and specify the template file for the registration page.

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

Configure the profile management experience and specify the template file for the profile management page.

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

Configure the data store where local identities are stored.

- Click **Configure Data Store** to begin.

Select a data store for customer identities

PingFederate stores customer identities in PingDirectory™.

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

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.

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

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.

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

Configure the mapping between the local identity profile fields and the data store attributes.

- Select an LDAP attribute under **Data Store Attribute** for each local identity field.

Review data store configuration

Review the data store configuration on the **Summary** screen.

- To make changes, click the heading over the information you want to edit, and then proceed with your changes.
- To keep the data store configuration, click **Done**.
- To restart the data store configuration process, click **Cancel**.

Review a local identity profile

Review a local identity profile and its configuration on the **Summary** screen.

- To make changes, click the heading over the information you want to edit, and then proceed with your changes.
- To keep the local identity profile, click **Done**, and then click **Save**.
- To restart the configuration process, click **Cancel**.

Configure the HTML Form Adapter for customer identities

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 397).

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

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:

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 *Set up PingDirectory for customer identities* on page 380.)
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 *Manage policy contracts* on page 258.)
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

Field	Data Store Attribute
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 381.)

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 *Configure the HTML Form Adapter for customer identities* on page 386.)

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** > **Contract Fulfillment** screen, fulfill the authentication policy contract with values from this local identity profile as follows:

Outbound Contract Fulfillment	Source	Value
subject	Local Identity	lipEmail
firstName	Local Identity	lipFirstName
lastName	Local Identity	lipLastName
mobileNumber	Local Identity	lipMobile

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.

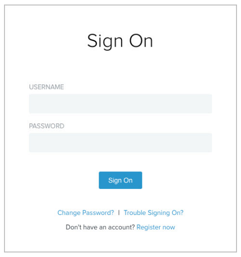
   Note: Other IdP authentication policies (if any) are enabled as well.

j) Click **Save** to keep your changes.

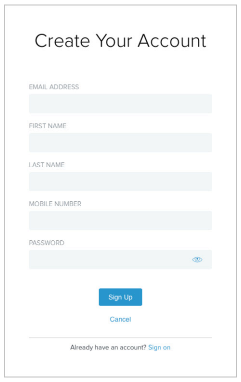
(For more information, see *Apply policy contracts or identity profiles to authentication policies* on page 240.)

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 343 and *Manage authentication policy contract grant mapping* on page 293).

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

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 386), you may skip to *step 3* to create a local identity profile.

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 *Set up PingDirectory for customer identities* on page 380.)
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 *Manage policy contracts* on page 258.)
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 386), 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 381.)

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 *Configure the HTML Form Adapter for customer identities* on page 386.)
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 386), 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** > **Contract Fulfillment** screen, fulfill the authentication policy contract with values from this local identity profile as follows:

Outbound Contract Fulfillment	Source	Value
subject	Local Identity	lipEmail
firstName	Local Identity	lipFirstName
lastName	Local Identity	lipLastName
mobileNumber	Local Identity	lipMobile

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**.
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** > **Contract Fulfillment** screen, fulfill the authentication policy contract with values from this local identity profile as follows:

Outbound Contract Fulfillment	Source	Value
subject	Local Identity	lipEmail
firstName	Local Identity	lipFirstName
lastName	Local Identity	lipLastName
mobileNumber	Local Identity	lipMobile

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.

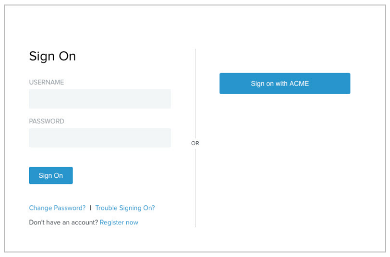
s) Click **Save** to keep your changes.

(For more information, see *Apply policy contracts or identity profiles to authentication policies* on page 240.)

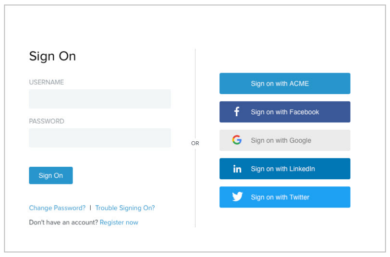
You have now successfully set up self-service registration with an option for users to register and subsequently authenticate via ACME. When users sign on through this HTML Form Adapter instance, they have two registration options:

- 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.

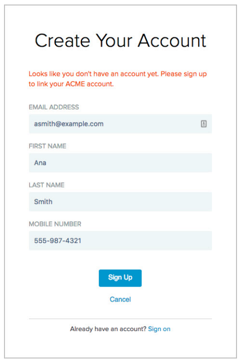
Based on the configuration of this sample use case, the following sign-on page is presented:


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:



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

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 386 or *Enable third-party identity providers* on page 389 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 386 and *Enable third-party identity providers* on page 389 as well. If you require more information for a given step, refer to the same step in one of the aforementioned pages.

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.

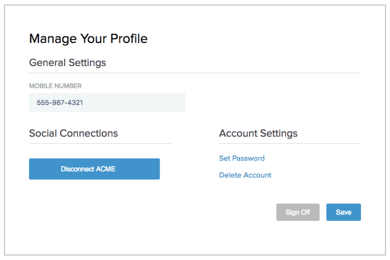
      Generally speaking, this is an optional feature. It is selected here because it is one of the requirements of this sample use case.

   g) Continue from step 3f as documented in both *Set up self-service registration* on page 386 or *Enable third-party identity providers* on page 389.

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.

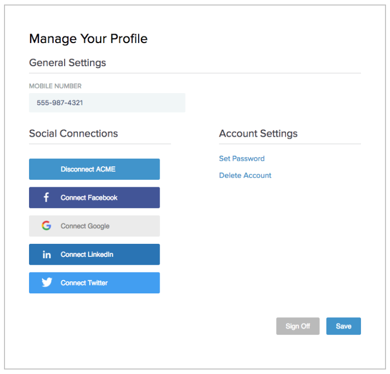
You have now successfully enabled profile management. Authenticated users can review and modify the local identity fields that have been configured to be shown on the profile management page and delete their local accounts if the option to do so has been enabled.

The following screen capture provides a sample of the profile management page based on the sample use case:



MOBILE NUMBER
555-987-4321

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:



MOBILE NUMBER
555-987-4321

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.

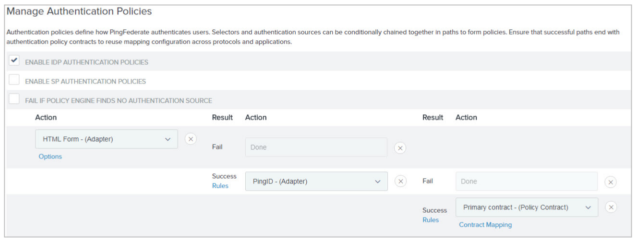
Create advanced registration mapping

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 386, and at the same time keep the policy that enforces the multifactor authentication requirement.

Configuration 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.

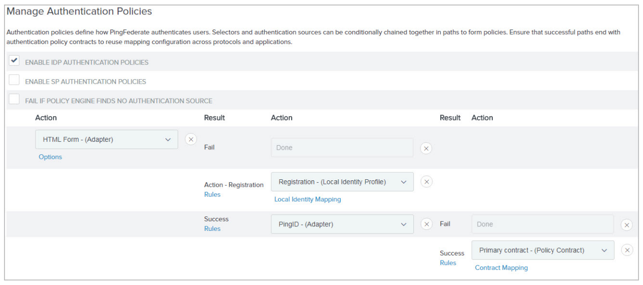
      If you want to enable profile management as well, select the relevant check box.
   b) Complete the rest of the configuration to create the local identity profile.
4. Configure the HTML Form Adapter instance for customer identities.
   a) On the **IdP Adapter** screen, select a local identity profile 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.

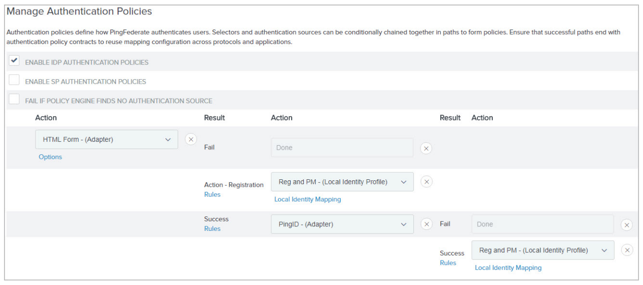
[✓] ENABLE IDP AUTHENTICATION POLICIES
[ ] ENABLE SP AUTHENTICATION POLICIES
[ ] FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

Action	Result	Action	Result	Action
HTML Form - (Adapter) Options	Fail	Done		
	Action - Registration Rules	Registration - (Local Identity Profile) Local Identity Mapping		
	Success Rules	PingID - (Adapter)	Fail	Done
			Success Rules	Primary contract - (Policy Contract) Contract Mapping

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) Options	Fail	Done		
	Action - Registration Rules	Reg and PM - (Local Identity Profile) Local Identity Mapping		
	Success Rules	PingID - (Adapter)	Fail	Done
			Success Rules	Reg and PM - (Local Identity Profile) Local Identity Mapping

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.

You have now successfully added the requested consumer registration (and profile management) use case to your current policy.

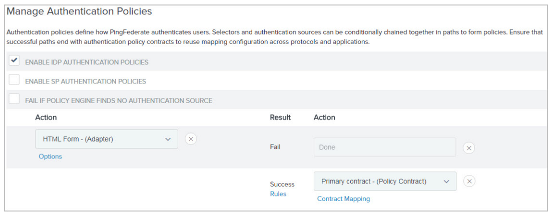
Enable third-party identity providers without registration

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:

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.

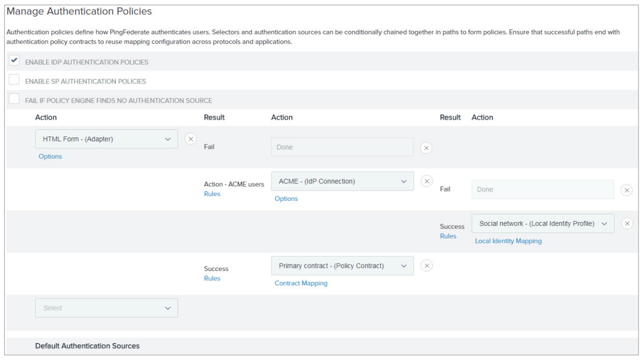
      When finished, click **Done**, which brings you back to the **Manage Authentication Policies** screen.
   c) For the **ACME users** path, select the IdP connection to ACME under **Action**.

      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**.
      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.

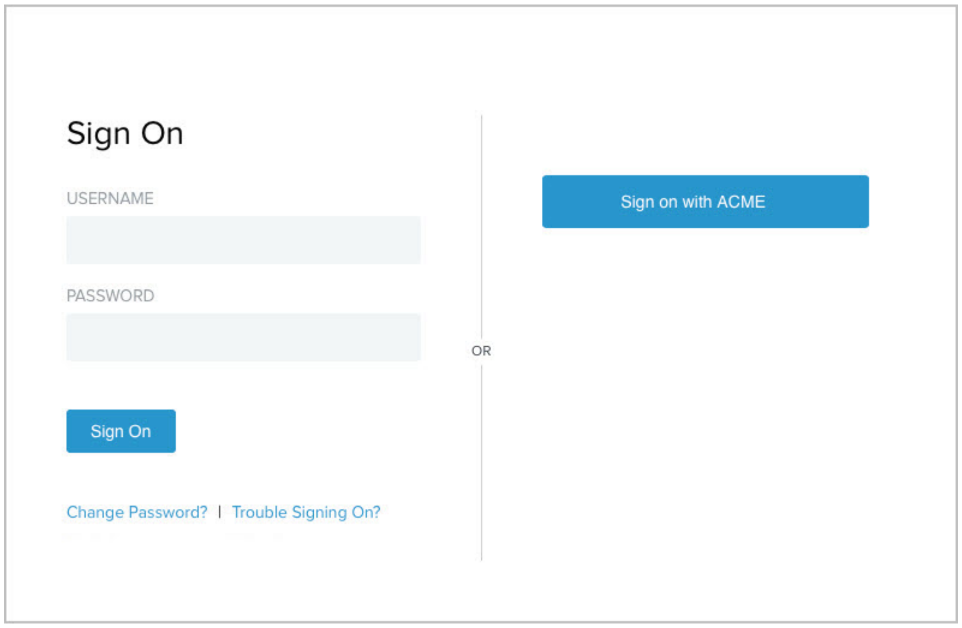
- [✓] ENABLE IDP AUTHENTICATION POLICIES
- [ ] ENABLE SP AUTHENTICATION POLICIES
- [ ] FAIL IF POLICY ENGINE FINDS NO AUTHENTICATION SOURCE

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				

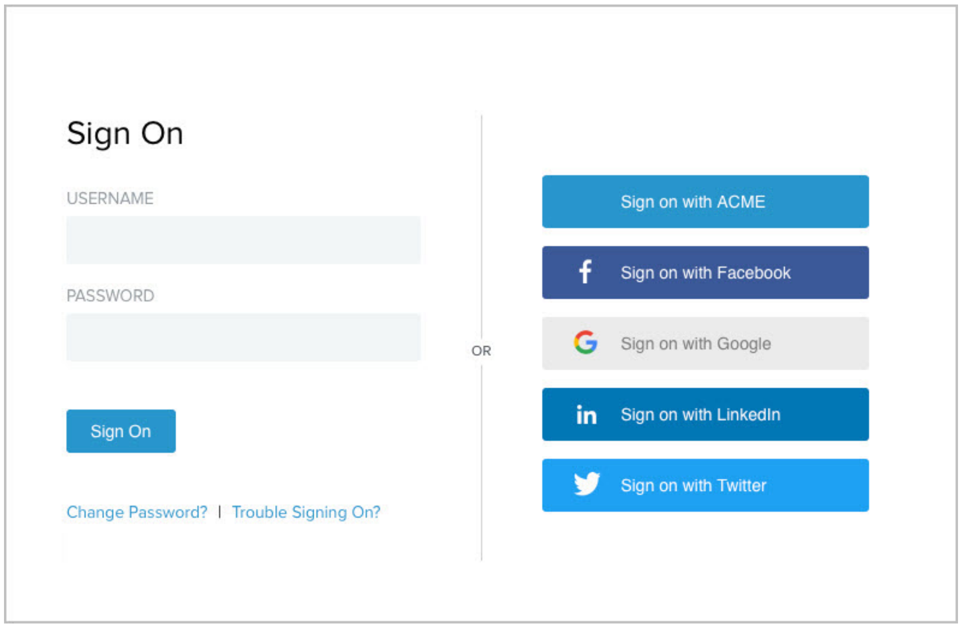
Default Authentication Sources

d) Click **Save** to keep your changes.

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 **Server Configuration** > **Server 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 **Server Configuration** > **Server 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 92).

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 86 and *Federation hub and authentication policy contracts* on page 90 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 470.

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 72).

Under **Application 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 **Application Integration** also include menu items for configuring plug-in **Token Generators** (see *Service provider STS configuration* on page 483).

Manage SP adapters

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.

- 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**.

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 **Server Configuration** > **Server Settings** > **System Options** screen.

Create an SP adapter instance

The first step in creating an adapter instance is choosing an adapter type.

- 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

Depending on the selected adapter, the **IdP Adapter** screen presents you with different configuration parameters.

- 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 the OpenToken SP Adapter* on page 517 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

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.

- Follow the on-screen instructions to complete the actions required.

Extend an SP adapter contract

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.

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Identify the target application

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 663.) Both fields are optional.

Note: If this is a child instance, select the override check box to modify the configuration.

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

1. On the **Summary** screen, review your configuration, modify as needed, and click **Done**.
2. 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**.

Configure target URL mapping

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.

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 532).

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 86).

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 420), PingFederate tries the next entry.

Field	Description
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.

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.

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 663*).

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 443 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 **Server Configuration** > **Server Settings** > **Roles & Protocol** 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 **Server Configuration** > **Server Settings** > **System Options** screen.

Create an Identity Store Provisioner instance

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.

Field descriptions

Field	Description
Instance Name	A descriptive name for the provisioner instance—for example, an identity management system name.
Instance ID	An internal identifier for the provisioner instance. Must be alphanumeric with no spaces.
Type	A list of deployed provisioner types available for creating a provisioner instance for the server. A developer typically deploys any new provisioner types before an administrator sets up a connection partner. The Identity Store Provisioner Type is limited to the provisioners currently installed on your server.
Parent Instance (Optional)	A list of configured instances for this type of provisioner. If applicable, select an instance that can be used as a basis for a parent/child configuration.

To define an identity store provisioner:

1. Click **Service Provider**.
2. Click **Identity Store Provisioners** under **Application Integration Settings**.
3. Click **Create New Instance on the Manage Identity Store Provisioners screen**.
4. Enter the **Instance Name** and **Instance ID** on the Type screen.
5. Select the **Type** from the drop-down menu.
6. 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.
7. Click **Next** and enter information on subsequent screens for this identity store setup, as indicated in the following sections.
8. Click **Done** on the **Summary** screen.
9. Click **Save** on the **Manage Identity Store Provisioners screen**.

Define the Identity Store Provisioner behavior

Different configuration parameters are available on the Identity Store Provisioner screen. These options are controlled by the provisioner plug-in software (see topics about identity store provisioner interfaces in the PingFederate *SDK developer's guide* for more information).

Extend the Identity Store Provisioner contract

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 Identity Store Provisioner 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.

To reach this screen for editing:

1. Click **Service Provider**.
2. Click **Identity Store Provisioners** under **Application Integration Settings**.
3. Click an instance name.
4. Click **Extended Contract** (if available).

To add an attribute:

Note: If this is a child instance, select the override check box to modify the configuration.

1. Enter the attribute name in the text box and click **Add**.
2. Click **Done** then click **Save** on the Manage Identity Store Provisioners screen.

Extend the Identity Store Provisioner contract for groups

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 Identity Store Provisioner 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.

To reach this screen for editing:

1. Click **Service Provider**.
2. Click **Identity Store Provisioners** under **Application Integration Settings**.
3. Click an instance name.
4. Click **Extended Group Contract** (if available).

To add an attribute:

Note: If this is a child instance, select the override check box to modify the configuration.

Enter the attribute name in the text box and click **Add**.

Click **Done** then click **Save** on the Manage Identity Store Provisioners screen.

Review the Identity Store Provisioner configuration

From the Summary screen, you can reach identity store provisioner instance settings for editing.

To edit the configuration:

1. Click the heading above the information you want to change.
2. Make your changes.
3. Click **Done** on the configuration page and **Save** on the Manage Identity Store Provisioners screen.

To save an identity store provisioner instance:

1. Click **Done** on the Summary screen.
2. Click **Save** on the Manage Identity Store Provisioners screen.

Configure default URLs

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 431).

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 *Localize messages for end users* on page 147. 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 532), 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 132).

View SP application endpoints

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate.

- 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 532 and *System-services endpoints* on page 542.

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 406).

You can view endpoints that your federation partners need to know to access your services under **Service Provider** > **Protocol Endpoints**.

Manage attribute requester mappings

If you are using the SAML 2.0 X.509 Attribute Sharing Profile (XASP), applications at your site must supply the subject distinguished name (DN) to identify a user's X.509 authentication certificate (see *Attribute Query and XASP* on page 47). 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 532).

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 **Server Configuration** > **Server 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.

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.

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 **Server Configuration** > **Server 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 154.

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 *Configure request parameters and SSO URLs* on page 465). For more information about Third Party Initiated Login flow, see the *OpenID Connect specification* (openid.net/specs/openid-connect-core-1_0.html#ThirdPartyInitiatedLogin).

Service	**URL and Description**
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.)
Metadata Service	`/` The default endpoint (empty path) from which partners can retrieve Auto-Connect metadata.
Assertion Consumer Service (SAML 1.x)	`/sp/acs.saml1` A SAML 1.x implementation URL that receives and processes assertions from an IdP.
Single Sign-on Service (WS-Federation)	`/sp/prp.wsf` The WS-Federation implementation URL that receives and processes security tokens and SLO messages.
WS-Trust STS (two endpoints)	`/sp/sts.wst` The SOAP endpoint that receives and processes SAML security-token requests from STS clients (web service providers at the SP site), validating SAML tokens or validating and exchanging SAML tokens based on the configured IdP connection. `/pf/sts.wst` Initiates direct STS token-to-token exchange and token validation, from an IdP token processor to an SP token generator, when that feature is configured (see *Token translator mappings* on page 494). **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 *Manage token generators* on page 483. (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 *Configure PingFederate properties* on page 98).

Virtual server ID support

For SAML connections using multiple virtual server IDs (see *Multiple virtual server IDs* on page 92), each virtual server ID has its own set of protocol endpoints. You may export a connection metadata for your partner on the **Server Configuration** > **Metadata Export** screen (see *Provide SAML metadata by file* on page 122).

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 542.)

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 *Federation planning checklist* on page 90).

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 419).

(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 433.)

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 *Manage data stores* on page 168).

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

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.

IdP Connections

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's connection ID, and default virtual server ID; narrow by protocol and status; and perform the following actions.

Create a connection

Follow the **Connection** wizard to create a new connection to your IdP partner.

Copy a connection

Follow the **Connection** wizard to create a new connection based on an existing (source) connection. This is most useful if the new connection and the source connection have many setting values in common. Note that the new connection stays disabled until you change its status.

Export a connection

Save the XML file as prompted. This is useful in situations where you want to make a backup of a connection prior to making changes to it.

Import a connection

Follow the **Import Connection** wizard to import a connection export. 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.

Export SAML metadata

Follow the **Export Metadata** wizard to generate a metadata for any SAML Browser SSO connection.

Update a SAML connection with metadata

Follow the **Import Metadata** wizard to update a SAML Browser SSO connection with metadata. 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.

Toggle the status of a connection

Click the toggle switch to enable or disable a connection.

Delete a connection

Delete a connection. You must click **Save** on the **IdP Connections** screen to confirm your request. If you change your mind, click **Undelete** to cancel your request.

Override logging mode

Override the verbosity of runtime transaction logging for all IdP connections.

Edit a connection

Open the connection by clicking on its name, select the setting that you want to reconfigure, and complete the change.

Resolve IdP connection errors

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.

- 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

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 **Server Configuration** > **Server Settings** > **Roles & Protcols** 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.

- 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 **Server Configuration** > **Server 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

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.

- 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 **Server Configuration** > **Server 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 47).

Import IdP metadata

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
- Enter a new metadata URL
- Select an existing metadata URL

Tip: Using a metadata URL streamlines the configuration process. For example, by importing a metadata URL, you can quickly establish a Browser SSO connection to an InCommon-participating partner (see *www.incommon.org/participants*).

When you enter a new URL or select an existing metadata URL, PingFederate also enables the automatic update option and checks the metadata daily. If PingFederate detects changes in the partner's digital signing certificates, encryption key, or contact information, it updates the connection automatically. If you prefer to update the connection manually, you can clear the **Enable Automatic Reloading** check box.

The frequency is configurable through the **Reload Delay** field on the **Server Configuration** > **Server Settings** > **Metadata Lifetime** screen.

Although optional, it is recommended that you turn on email notifications for SAML metadata update events on the **Server Configuration** > **Server Settings** > **Runtime Notification** screen.

Note: If the metadata contains changes that require additional configuration, the email 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: These capabilities also lower the cost of maintaining connections with InCommon participants.

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	1. On the **Import Metadata** screen, select the **File** option. 2. Choose the metadata file, and then click **Next**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note:** If the metadata file is digitally signed but the verification certificate is provided outside of the metadata, import the metadata

Metadata medium	Steps
	verification certificate on the **Import Certificate** screen, and then click **Next**. 3. On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. 4. Click **Next**.
A new metadata URL	1. On the **Import Metadata** screen, select the **URL** option. 2. Enter a new metadata URL. **Note:** If you specify a URL that has already been entered into the system through the **Server Configuration** > **Metadata URLs** configuration wizard, the administrative console reminds you to select it from the **Existing URL Name** list. 3. Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. 4. Click **Load Metadata**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note:** If 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**. 5. On the **Metadata Summary** screen, review the signature information to evaluate the authenticity of the metadata. 6. Click **Next**.
An existing metadata URL	1. On the **Import Metadata** screen, select the **URL** option. 2. Select an existing metadata from the list, and then verify the URL in the **Selected Existing URL**. 3. Optionally, clear the **Enable Automatic Reloading** check box to disable automatic update. 4. Click **Load Metadata**. **Note:** If the metadata contains multiple entries, select the desired partner from the **Select Entity ID** list and click **Next**. **Note:** If there is a digital signature error, correct the issue using the **Server Configuration** > **Metadata URLs** configuration wizard. 5. Click **Next**.

Identify the partner

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.

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.
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 **Server Configuration** > **Server 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.

Field	Description
	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.)

Define additional issuers

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.

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 Browser SSO

Browser-based SSO (also known as Browser SSO) is another term for secure SSO, which relies on a user's web browser and HTTP to broker SSO tokens (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**.

SP Browser SSO configuration steps

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 33). 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 SSO steps

- *Select SAML profiles* on page 417

- *Configure user-session creation* on page 417
    - *Choose an identity mapping method* on page 417
    - *Define an attribute contract* on page 418
    - *Manage target session mappings* on page 419
- *Manage protocol settings* on page 424
    - *Specify SSO service URLs (SAML)* on page 425
    - *Define SLO service URLs (SAML 2.0)* on page 427
    - *Select allowable SAML bindings (SAML)* on page 427
    - *Specify an artifact lifetime (SAML 2.0)* on page 427
    - *Define artifact resolver locations (SAML)* on page 428
    - *Configure default target URLs* on page 431
    - *Override authentication context in an IdP connection* on page 431
    - *Configure signature policy* on page 432
    - *Specify XML encryption policy (for SAML 2.0)* on page 433

WS-Federation SSO steps

- *Configure user-session creation* on page 417
    - *Choose an identity mapping method* on page 417
    - *Define an attribute contract* on page 418
    - *Manage target session mappings* on page 419
- *Manage protocol settings* on page 424
    - *Specify a service URL (WS-Federation)* on page 426
    - *Configure default target URLs* on page 431
    - *Configure signature policy* on page 432

SAML 1.x SSO steps

- *Select SAML profiles* on page 417
- *Configure user-session creation* on page 417
    - *Choose an identity mapping method* on page 417
    - *Define an attribute contract* on page 418
    - *Manage target session mappings* on page 419
- *Manage protocol settings* on page 424
    - *Specify SSO service URLs (SAML)* on page 425
    - *Select allowable SAML bindings (SAML)* on page 427
    - *Define artifact resolver locations (SAML)* on page 428
    - *Configure default target URLs* on page 431
    - *Configure signature policy* on page 432

OpenID Connect SSO steps

- *Configure user-session creation* on page 417
    - *Choose an identity mapping method* on page 417
    - *Define an attribute contract* on page 418
    - *Manage target session mappings* on page 419
- *Manage protocol settings* on page 424
    - *Configure OpenID Provider information* on page 428
    - *Configure default target URLs* on page 431

- *Override authentication context in an IdP connection* on page 431

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 454). 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

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 35 and *SAML 2.0 profiles* on page 38.)

Note that the **SAML Profiles** screen does not apply to OpenID Connect and WS-Federation IdP connections.

- 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

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 *Federation hub use cases* on page 86 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.

- To continue, click **Configure User-Session Creation**.

Choose an identity mapping method

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 *Identity mapping* on page 78). 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 *Federation planning checklist* on page 90).

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 *User attributes* on page 79).

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.

- 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 *Configure just-in-time provisioning* on page 434).
- 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 *Define an account-linking data store* on page 177.

- 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 289).

Define an attribute contract

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 79). 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 83).

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.

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.

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

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 222.

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 86.

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.

- 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**.

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 402).

Select a target session

The first task of the mapping configuration is to map an adapter instance or an authentication policy contract to your connection.

- 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 77.

- 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.

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

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.

- 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 *Manage SP adapters* on page 401).

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

When you multiplex one connection for multiple environments (see *Multiple virtual server IDs* on page 92), 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.

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.

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

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 222.

- 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 605.

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

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 86).

At runtime, an SSO operation fails if PingFederate cannot fulfill the required attribute.

- 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 Expiration** setting.

    - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Custom**

  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 *Attribute mapping expressions* on page 593.
- **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.

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

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 *Multiple virtual server IDs* on page 92), 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 *Issuance criteria and multiple virtual server IDs* on page 596).

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.

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 *Choose an identity mapping method* on page 417).
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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the target session mapping

When you have finished adding a new or modifying an existing instance of an adapter or an authentication policy contract, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review the session creation summary

When you are finished with the **User-Session Creation** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Manage protocol settings

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

- To continue, click **Configure Protocol Settings**.

Specify SSO service URLs (SAML)

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.

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 593. 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 598.
   d) Click **Add**.
   e) Optional: Repeat to add another message customization.

Specify a service URL (WS-Federation)

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 189).

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.
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.

Define SLO service URLs (SAML 2.0)

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.

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.

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)

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.

- 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.

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)

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.

- Optional: Override the default value of the **Artifact Lifetime** field.

  The default value is 60 (seconds).

If you are editing an existing connection, you can update the artifact lifetime.

Define artifact resolver locations (SAML)

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.

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.

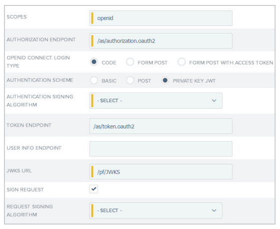
2. Optional: Enter your partner's source ID.

   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.

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

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**.
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 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 *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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.

Field	Description
	Changes made in the static-key configuration may affect runtime transactions and require additional changes here. For more information, see *Manage keys for OAuth and OpenID Connect* on page 205. **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'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 **Server Configuration** > **Server 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 Connect Metadata endpoint to obtain the information (see *OpenID Provider configuration endpoint* on page 564).
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 integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see *Supported hardware security modules* on page 56.) 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.
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 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

Field	Description
	HSM integration and static keys, see *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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 *Manage keys for OAuth and OpenID Connect* on page 205. **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 **Server Configuration** > **Server 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 Connect Metadata endpoint for it (see *OpenID Provider configuration endpoint* on page 564).

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 request parameters and SSO URLs* on page 465).

Configure default target URLs

Use the **Protocol Settings** > **Overrides** screen to assign a default target URL for this IdP connection.

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

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.

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

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

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 policy coordination* on page 75).

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.)

- To continue, select the option (or options) based on your partner agreement.

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)

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.

If the name identifier (or SAML_SUBJECT) of an assertion is encrypted, you and your partner may also want to encrypt the identifier in subsequent SAML 2.0 SLO messages (if you are using an SLO profile).

Note that "The entire assertion" selection on the Encryption Policy screen includes the SAML_SUBJECT and all attributes.

To enable encryption:

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.

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

When you are finished with the **Protocol Settings** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review Browser SSO settings

When you are finished with the **Browser SSO** primary task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

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 *Choose IdP connection options* on page 412). 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 *Attribute Query and XASP*).

Set the Attribute Authority Service URL

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*).

To configure the URL:

- Enter the fully qualified URL or a relative path if you have defined a base URL (see *Identify the partner* on page 413).

Map attribute names for Attribute Query

If the application at your site uses different names for user attributes than the names defined by the Attribute Authority, 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.

To map attributes:

1. Enter the Local Name and Remote Name of an attribute and click **Add**.

   Repeat this step for all attributes requiring mapping.
2. Click **Next**.

To edit a mapping:

1. Click **Edit** under Action for the mapping.
2. Make your change(s) and click **Update**.

   Note: If you change your mind, ensure that you click the **Cancel** *link* in the Actions column, not the **Cancel** *button*, which discards any other changes you might have made in this configuration.
3. Click **Done** and then **Save** on the Attribute Query screen.

Configure security policy for Attribute Query

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 *Configure security credentials* on page 454).

This screen also allows you to mask incoming attribute values in log files (see *Attribute masking* on page 83). When you enable this selection, all user attributes returned from this IdP are masked.

To configure attribute-query security policy for this partner:

- Check or clear the check boxes and click **Next** or **Done**.

Review the Attribute Query settings

On the Summary screen you can review the Attribute Query configuration.

To reconfigure saved profiles:

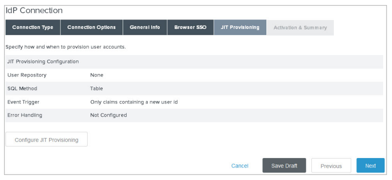
1. Click the heading over the information you want to change.
2. Click **Done** on the screen containing your change.

   If you need to make additional changes, do so and continue by clicking **Done** until you reach the Attribute Query screen.
3. Click **Save** on the Attribute Query screen.

Configure just-in-time provisioning

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.

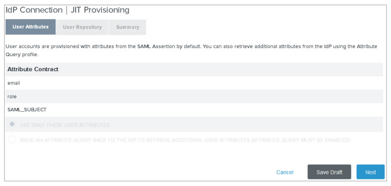
JIT Provisioning Configuration	
User Repository	None
SQL Method	Table
Event Trigger	Only claims containing a new user id
Error Handling	Not Configured

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)

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.


IdP Connection | JIT Provisioning

User accounts are provisioned with attributes from the SAML Assertion by default. You can also retrieve additional attributes from the IdP using the Attribute Query profile.

Attribute Contract
email
role
SAML_SUBJECT

● USE ONLY THESE USER ATTRIBUTES

○ ISSUE AN ATTRIBUTE QUERY BACK TO THE IDP TO RETRIEVE ADDITIONAL USER ATTRIBUTES (ATTRIBUTE QUERY MUST BE ENABLED)

Note: 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.

- If you and your IdP partner have agreed to use the Attribute Query profile for provisioning, select that option before leaving this screen.

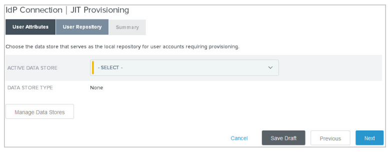
  You configure the Attribute Query profile later in the task flow.

Identify the user repository

PingFederate's JIT Provisioning currently supports LDAP v3-compliant user stores and the JDBC-compliant Microsoft SQL Server.

Note: We recommend using the latest version of the Microsoft SQL Server JDBC Driver (version 4.1 or higher), which is compatible with current and recent Server JRE versions.

- 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 -

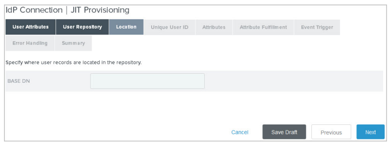
DATA STORE TYPE: None

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 435
  - *Enter an LDAP filter* on page 436
  - *Identify provisioning attributes for LDAP* on page 436
- If you are using a Microsoft SQL Server, skip to this section:
  - *Choose a SQL method*

Specify an LDAP user-record location

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.



Specify where user records are located in the repository.

BASE DN

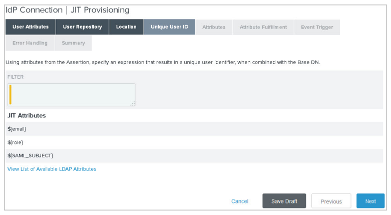
Note: This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

- 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

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}

Note: This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

- 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.

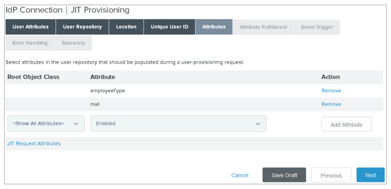
  The left-side variable is an attribute in your user-data store. Click the link near the lower-left corner of the screen to see a list of available attributes.

  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

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.

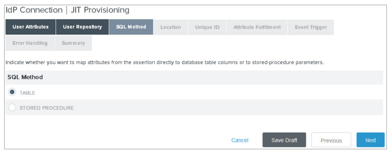
Root Object Class	Attribute	Action
	employeeType	Remove
	mail	Remove
<Show All Attributes>	Enabled	

Note: This screen appears only when an LDAP data store is chosen on the **User Repository** screen.

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

For JDBC data stores, PingFederate allows you to map attributes directly to a single database table (the default) or to SQL stored-procedure parameters.


IdP Connection | JIT Provisioning

Indicate whether you want to map attributes from the assertion directly to database table columns or to stored-procedure parameters.

SQL Method

[✓ TABLE] STORED PROCEDURE

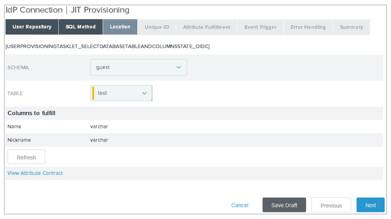
Note: This screen appears only when a Microsoft SQL Server data store is chosen on the **User Repository** screen.

- 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 437
  - *Specify a unique-ID database column* on page 437
- If you are using a stored procedure, skip to this section:
  - *Specify a stored-procedure location* on page 438

Specify a database user-record location

For database provisioning to a table, indicate where PingFederate should write new user records or update existing ones.


IdP Connection | JIT Provisioning

[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.

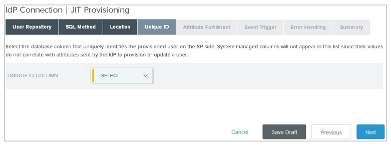
- Select the database schema and the table.

Field	Description
Schema	Select the table structures that store information within the database.
Table	Select the name of the database table that contains user records.
Columns to fulfill	For the selected table, all attributes and their data types are displayed. If the list of columns is not 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

PingFederate uses the column specified on this screen to check whether a user record already exists for the incoming SSO token.



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.

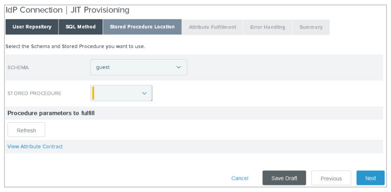
UNIQUE ID COLUMN: - SELECT -

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.

- Select a column that represents a unique characteristic about the database entry for a particular user.

Specify a stored-procedure location

If you are using a stored procedure for provisioning the user database, specify its location on this screen.



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.

- 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

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.

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 **Server Configuration** > **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.

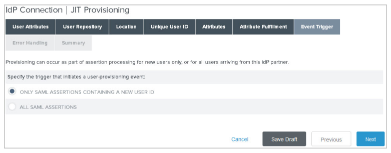
   > **Note:** For Active Directory, enter `user` in the text box for objectClass. For the Oracle Directory Server, enter `inetOrgPerson`.
3. Optional: When mapping to a Microsoft SQL Server data store, test the insertion.

Table	1. Click **Test insert into '*table*'**. 2. Enter values for each applicable target parameter. 3. 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	1. Click **Test call to '*procedure*'**. 2. Enter values for each applicable target parameter. 3. Click **Test Stored Procedure Call**. For stored procedures, only a confirmation 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

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.



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]

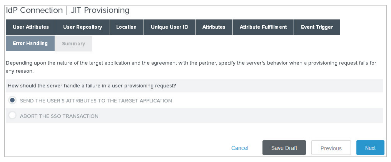
ALL SAML ASSERTIONS

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

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.



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

The **Summary** screen provides an overview of your provisioning configuration.

1. Click the heading over the information you want to change.
2. Click **Done** on the screen containing your change.

   If you need to make additional changes, do so and continue by clicking **Done** until you reach the **JIT Provisioning** screen.
3. Click **Save** (or **Next**) on the **Inbound Provisioning** screen.

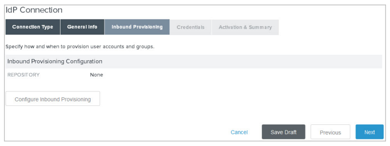
Configure SCIM inbound provisioning

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 445). 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 447).

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.

1. If you have not already done so, enable the **Inbound Provisioning** option (under the Server Provider role) on the **Server Configuration** > **Server 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.



Specify how and when to provision user accounts and groups.

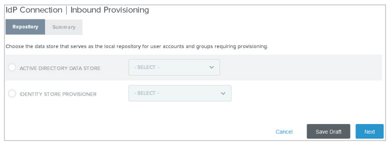
Inbound Provisioning Configuration

REPOSITORY None

Specify the user repository

PingFederate currently supports AD user stores (requires an LDAPS connection to the data store) and custom identity store provisioners for inbound provisioning.

- Choose either the **Active Directory Data Store** or **Identity Store Provisioner** option and then specify the data store from the list.

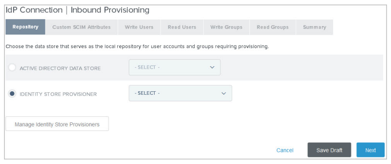


Choose the data store that serves as the local repository for user accounts and groups requiring provisioning.

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.



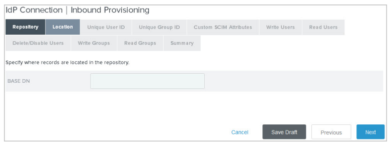
Choose the data store that serves as the local repository for user accounts and groups requiring provisioning.

ACTIVE DIRECTORY DATA STORE - SELECT -

● IDENTITY STORE PROVISIONER - SELECT -

Identify an LDAP user-record location

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.

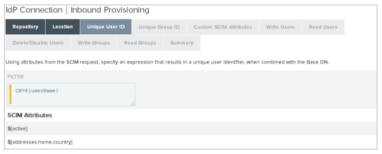
BASE DN

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

- 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

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.

- 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.

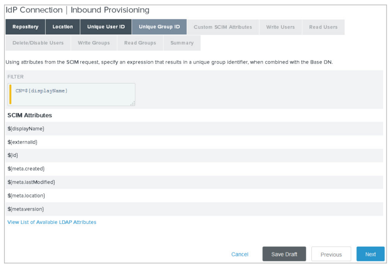
  The left-side variable is an attribute in your user-data store. Click the link near the lower-left corner of the screen to see a list of available attributes.

  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

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.



Using attributes from the SCIM request, specify an expression that results in a unique group identifier, when combined with the Base DN.

FILTER

`CN=${displayName}`

SCIM Attributes

${displayName}

${externalId}

${id}

${meta.created}

${meta.lastModified}

${meta.location}

${meta.version}

Note: This 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.

- 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

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.

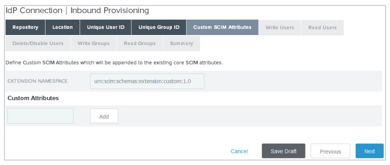
```
{
  "userName":"CBrown",
  "active":true,
  "schemas":[
    "urn:scim:schemas:core:1.0",
    "urn:scim:schemas:extension:custom:1.0"
  ],
  ...
  "urn:scim:schemas:extension:custom:1.0":{
    "supervisor":"JSmith",
    "territories":[
      "Montana",
      "Idaho",
      "Wyoming"
    ],
    "options":{
      "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.

Attribute Name	Attribute Type	Sub-Attributes (Complex)
		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*.

- To specify a schema extension, enter the URI of the schema extension in the **Extension Namespace** field.



IdP Connection | Inbound Provisioning

Define Custom SCIM Attributes which will be appended to the existing core SCIM attributes.

EXTENSION NAMESPACE: urn:scim:schemas:extension:custom:1.0

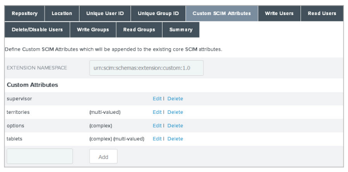
Custom Attributes

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

The administrative console displays the **Custom SCIM Attribute Options** screen, where you may

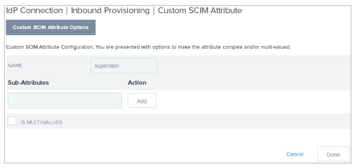
- Change the attribute name
- Set the attribute as a simple multivalued attribute
- Add sub-attributes to make it a complex attribute
- Add sub-attributes and set the attribute as a complex multivalued attribute

(For more information, see *Configure custom SCIM attribute options* on page 444.)

Configure custom SCIM attribute options

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.

- 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

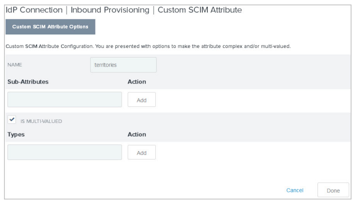
Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: supervisor

Sub-Attributes | Action

☐ 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
Add
Add
Cancel
Done

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: territories

Sub-Attributes	Action

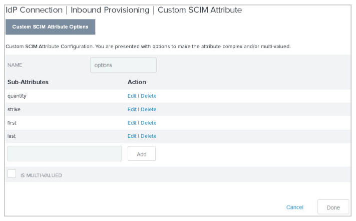
[✓] IS MULTI-VALUED

Types	Action

- 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 Options
Add
Cancel
Done

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

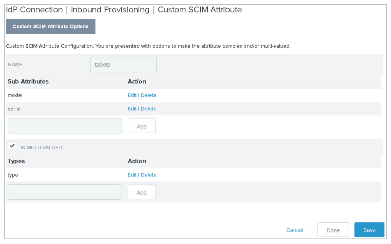
[ ] IS MULTI-VALUED

- 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 441), 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 Options
Add
Add
Cancel
Done
Save

Custom SCIM Attribute Configuration. You are presented with options to make the attribute complex and/or multi-valued.

NAME: tablets

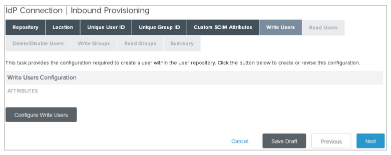
Sub-Attributes	Action
model	Edit I Delete
serial	Edit I Delete

[✓] IS MULTI-VALUED

Types	Action
type	Edit I Delete

Write user information to the data store

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.

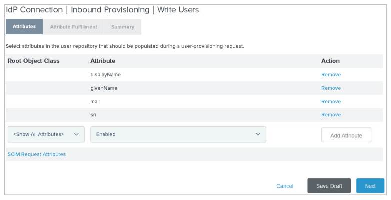


This task provides the configuration required to create a user within the user repository. Click the button below to create or revise this configuration.

- Click **Configure Write Users** to continue.

Identify inbound provisioning attributes for LDAP

On the **Attributes** screen, select the data-store attributes you want to provision.


IdP Connection | Inbound Provisioning | Write Users

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.

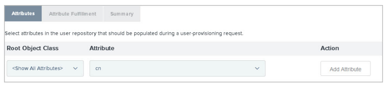
Several attributes are managed internally by PingFederate and do not require mapping:

- 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.

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.

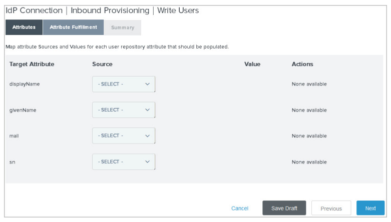
   Select attributes in the user repository that should be populated during a user-provisioning request.

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

Use this screen to map attribute values in the SCIM request to user-account attributes.


IdP Connection | Inbound Provisioning | Write Users

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

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 Expiration** setting.

     - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Server Configuration** > **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**.

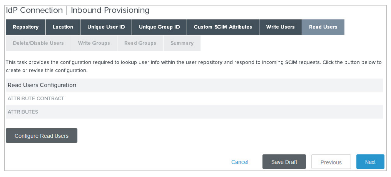
Review user mapping (Write Users) configuration

The **Summary** screen provides an overview of the inbound provisioning configuration for request mapping.

- When you finish with a new configuration, click **Done**.

Configure a SCIM response

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.



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

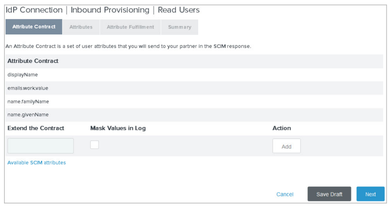
ATTRIBUTE CONTRACT

ATTRIBUTES

- Click **Configure Read Users** to continue.

Identify expected user attributes for the SCIM response

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**.



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

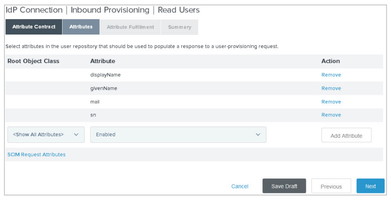
- To add an attribute, enter the attribute name in the text box, select the check box under **Mask Values in Log** as needed, and click **Add**.

  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

Select the LDAP attributes you want to map to attributes in the SCIM response.



Select attributes in the user repository that should be used to populate a response to a user-provisioning request.

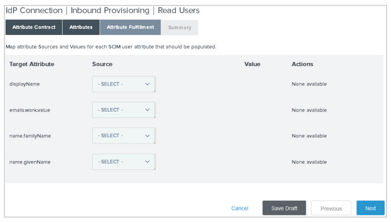
Root Object Class	Attribute	Action
	displayName	Remove
	givenName	Remove
	mail	Remove
	sn	Remove
<Show All Attributes>	Enabled	Add Attribute

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

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

Use this screen to map outgoing user-account attributes to SCIM responses to READ requests.



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

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 Expiration** setting.
>
> - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Server Configuration** > **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

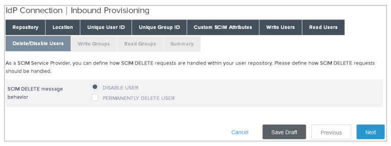
The **Summary** screen provides an overview of the inbound provisioning configuration for SCIM response mapping.

- When you finish with a new configuration, click **Done**.

Configure the handling of SCIM delete requests

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.

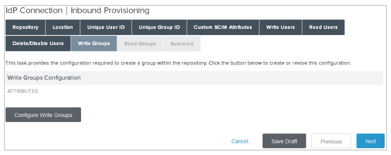
- Select **Disable User** to make the user inactive within the data store. This approach might be preferred in situations where accounts must be retained for auditing reasons.

  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

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.

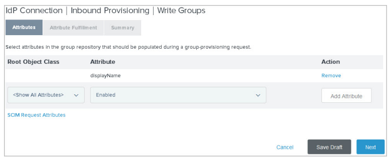
Write Groups Configuration

ATTRIBUTES

- Click **Configure Write Groups** to continue.

Identify inbound provisioning group attributes for LDAP

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:

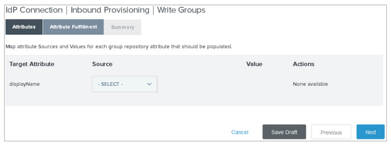
- objectClass
- objectGUID
- member

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.

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

Use this screen to map attribute values in the SCIM request to group attributes.



Map attribute Sources and Values for each group repository attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available

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 Expiration** setting.

     - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Server Configuration** > **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**.

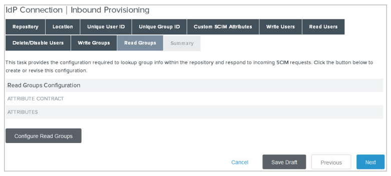
Review group mapping (Write Groups) configuration

The **Summary** screen provides an overview of the inbound provisioning configuration for request mapping.

- When you finish with a new configuration, click **Done**.

Configure a SCIM response for groups

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

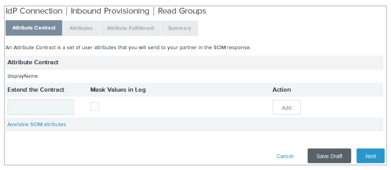
ATTRIBUTE CONTRACT

ATTRIBUTES

- Click **Configure Read Groups** to continue.

Identify expected group attributes for the SCIM response

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

- 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**.

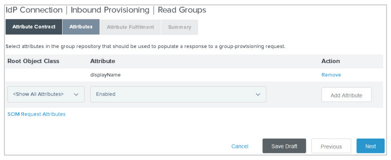
  Attribute names are case-sensitive and must correspond to the attribute names expected by your partner. To see a list of available attributes, click **Available SCIM attributes**.

- 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

Select the LDAP attributes you want to map to attributes in the SCIM response.



Select attributes in the group repository that should be used to populate a response to a group-provisioning request.

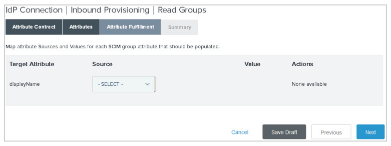
Root Object Class	Attribute	Action
	displayName	Remove
<Show All Attributes>	Enabled	Add Attribute

Note: This 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.

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

Use this screen to map outgoing group attributes to SCIM responses to READ requests.



Map attribute Sources and Values for each SCIM group attribute that should be populated.

Target Attribute	Source	Value	Actions
displayName	- SELECT -		None available

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 Expiration** setting.

     - To set lifetime based on the per-client **Persistent Grants Expiration** 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 Expiration** 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 **Server Configuration** > **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

The **Summary** screen provides an overview of the inbound provisioning configuration for SCIM response mapping.

- When you finish with a new configuration, click **Done**.

Review the inbound provisioning configuration

On the **Summary** screen you can review the **Inbound Provisioning** configuration.

1. Click the heading over the information you want to change.
2. Click **Done** on the screen containing your change.

   If you need to make additional changes, do so and continue by clicking **Done** until you reach the **Inbound Provisioning** screen.
3. Click **Save** (or **Next**) on the **Inbound Provisioning** screen.

Configure security credentials

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. Your connection configuration may involve any or all of the following:

- *Manage back-channel authentication* on page 455
- *Manage digital signature settings* on page 457
- *Manage signature verification settings* on page 457

- *Choose an encryption certificate (SAML 2.0)* on page 458
- *Choose a decryption key (SAML 2.0)* on page 459

- To continue, click **Configure Credentials**.

Manage back-channel authentication

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.

Refer to the following topics for configuration steps:

- *Configure back-channel authentication for outbound messages* on page 455
- *Configure back-channel authentication for inbound messages* on page 456

Configure back-channel authentication for outbound messages

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 *Manage SSL client keys and certificates* on page 196).

   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 for outbound authentication requirements:
   - If you need to make any changes, click the heading over the information you want to edit.
   - If you are creating a new connection and want to keep your configuration, click **Done**.
   - If you are editing an existing configuration and want to keep your changes, click **Save**.

Configure back-channel authentication for inbound messages

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 **Server Configuration** > **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 *Digital signing policy coordination* on page 75.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: 1. Enter the Subject DN of the certificate. 2. (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.

Trust model	Subsequent steps
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 *Manage certificates from partners* on page 204).

4. On the **Summary** screen for inbound authentication requirements:

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Manage digital signature settings

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.)

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 198).

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

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.

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 **Server Configuration** > **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 75.)

Trust model	Subsequent steps
Anchored	On the **Subject DN** screen: 1. 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. 2. (Optional) Select the **Restrict Issuer** check box and enter the Issuer DN of the certificate. Alternatively, extract it from your partner's certificate. **Important:** Consider enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.
Unanchored	On the **Signature Verification Certificate** screen: 1. Select a primary 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 204). 2. Select a secondary certificate from the list. **Tip:** Use this field if your partner has sent you a new certificate to replace one that is ready to expire. The server will automatically verify against the secondary certificate when the primary one expires.

3. On the **Summary** screen for signature verification:

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Choose an encryption certificate (SAML 2.0)

If SAML_SUBJECT is encrypted, either by itself or as part of a whole assertion, then all references to this name identifier in SAML 2.0 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 433.

You must also choose a certificate if encryption of the name identifier is required for an Attribute Request profile.

1. Optional: Change the default settings under **Block Encryption Algorithm**.

   Due to import control restrictions, the standard JRE distribution supports strong but not unlimited encryption. To use the strongest AES encryption, when permissible, download and install the appropriate version of "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files" from the *Oracle Downloads* website (www.oracle.com/technetwork/java/javase/downloads/index.html).

   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/)

   Note: Under **Key Transport Algorithm**, **RSA-v1.5** is disabled for new connections for security reasons. If you are updating an existing connection that uses RSA-v1.5, consider changing the selection for increased security.

Note that the JCE provider libraries distributed with Gemalto SafeNet Luna SA HSM does not support the RSA-OAEP algorithm at the time of the PingFederate 7.1 R2 release. As a result, RSA-v1.5 is still allowed when deploying PingFederate with a Gemalto SafeNet Luna SA HSM.

2. 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 204).

Choose a decryption key (SAML 2.0)

As part of XML encryption, you must identify a certificate and key for PingFederate to use to decrypt incoming assertions or assertion elements (see *Specify XML encryption policy (for SAML 2.0)* on page 433).

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 *Manage digital signing certificates and decryption keys* on page 198).
2. Optional: Select the secondary XML decryption key from the list.

Review credential settings

When you finish configuring various settings under the **Credentials** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review an IdP connection

When you finish creating or modifying a connection, you can review the connection settings and toggle the connection status on the **Activation & Summary** screen.

Important: When creating a new connection, the default connection status is **Inactive** when you reach the **Activation & Summary** screen.

Regardless of whether you choose to activate 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 (underneath the **Connection Status** field) is a sample URL that the developers of the target application might use to invoke SSO for the connection (see *View SP application endpoints* on page 406).

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.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, you must click **Save**.
- If you are modifying an existing connection (including toggling the connection status) and want to keep your changes, you must click **Save**.

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

1. If you have not already done so, enable the OpenID Connect protocol for IdP connections.
   a) Go to the **Server Configuration** > **Server 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.

    (You may also map an authentication policy contract if your PingFederate server is a federation hub that bridges an OP to an SP.)

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.

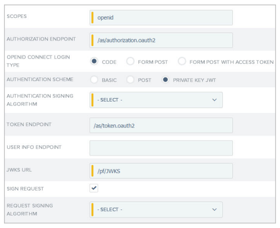
If your target application requires the associated access token, select **Context** as the source and **Access Token** as the value.

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`.

Field	Description
	• If the client is configured to support the Implicit Client profile *and* the target application requires the associated access token, select **Form POST with access token**. The resulting values of the response_type parameter are `id_token token`. The default selection (without loading metadata from the OP) is **Code**.
Authentication Scheme	The client authentication method that PingFederate uses. Applicable and visible only to clients supporting the Basic Client profile. • Select **Basic** to submit credentials 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 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 *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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 *Manage keys for OAuth and OpenID Connect* on page 205. **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'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 **Server Configuration** > **Server 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 Connect Metadata endpoint to obtain the information (see *OpenID Provider configuration endpoint* on page 564).
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**.)

Field	Description
	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 integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see *Supported hardware security modules* on page 56.) 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.
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 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 *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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 *Manage keys for OAuth and OpenID Connect* on page 205. **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 **Server Configuration** > **Server 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 Connect Metadata endpoint for it (see *OpenID Provider configuration endpoint* on page 564).

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 *Configure request parameters and SSO URLs* on page 465).
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** near the top of the **Summary & Activation** screen is a sample URL that developers of the target application might use to invoke SSO for the connection. Additional query parameters might also be sent to the `/sp/startSSO.ping` endpoint.

    Note: If you have selected the **No Mapping** option on the **Identity Mapping** screen (because you are passing end-user claims to the target application through an authentication policy contract in an SP authentication policy), 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 465.

Configure request parameters and SSO URLs

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:

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.

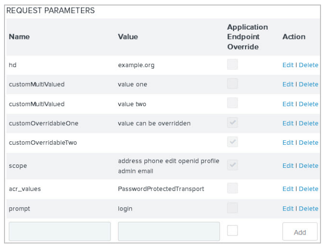
   If the target application does not provide the parameter in its SSO URL, the default value (if any) is used.

   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.

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 vs. 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 Auto-Connect

When your IdP partner is also using PingFederate 5 or higher (or is otherwise able to provide interoperable SAML 2.0 metadata via HTTP on demand), you may choose to use Auto-Connect™ for that partner (see *Using Auto-Connect*). This configuration can be shared among an unlimited number of SAML 2.0 partners.

Note: You enable the SAML 2.0 Auto-Connect profile under System Settings (see *Choose roles and protocols* on page 161).

Once Auto-Connect™ is enabled on your PingFederate server, complete the configuration from the Service Provider menu. This configuration entails:

- Setting up a common connection for all Auto-Connect partners
- Establishing a list of IdP partner domains authorized to use the connection

Initial setup for IdP Auto-Connect

The basic configuration for SP Auto-Connect™ requires only:

- Choosing a signing certificate for authentication requests and other SAML messages
- Configuring user-session creation information

All other partner-connection specifications are handled automatically at runtime.

Choose a certificate for IdP Auto-Connect

For Auto-Connect™ runtime processing, authentication requests and SLO messages must be signed, because they are sent over either the POST or redirect bindings (see *SAML 2.0 profiles* on page 38).

Note: The signing certificate is embedded in your server's Auto-Connect metadata (see *Auto-Connect* on page 94); there is no need to exchange certificates with your partners.

You can use the same certificate used for signing metadata (see *Configure metadata signing* on page 166). If you use a different certificate, ensure that it meets Auto-Connect validation requirements (see *Auto-Connect security model* on page 95).

To specify a certificate:

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 198).
2. 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 user-session creation for IdP Auto-Connect

Configuring user-session creation for Auto-Connect™ is similar to configuring the same settings for regular partner connections.

- Click **Configure User-Session Creation** to continue.

  For configuration information, refer to sections under *Configure user-session creation* on page 417.

  Note: Attributes sent from the IdP via Auto-Connect are passed to your applications, regardless of whether they are listed in the attribute contract (see *Define an attribute contract* on page 418).

Review IdP Auto-Connect settings

When you finish configuring your IdP Auto-Connect™ initial setup, you may choose to activate the common connection immediately on the Activation & Summary screen. (No runtime processing occurs until your partner's Auto-Connect gateway is also established and a user initiates an SSO or SLO event.)

Important: Regardless of whether you choose to activate a newly configured connection now or later, you must click **Save** on the Activation & Summary screen if you want to keep the configuration.

You can deactivate the connection at any time (for maintenance, for example). While a connection is inactive, all SSO or SLO transactions to or from Auto-Connect partners are disabled.

To change a Connection Status:

- Select Active or Inactive and then click **Save**.

To modify a setting:

1. Locate the currently configured setting under one of the summary headings and then click the subheading above the data.

   Note: Changes made to Auto-Connect settings will be out of sync, temporarily, with metadata caches that any currently active partners might be using. If your connection is in production, you might wish to lower your server's metadata lifetime in advance of making configuration changes (see *Configure metadata lifetime* on page 166).

2. Change the information and click **Save**, if available.

   If **Save** is not available, additional, dependent changes are required; click **Next** or **Done** until you reach a screen containing a **Save** button. Then click **Save** and continue as needed.

Specify allowed IdP domains

This screen provides PingFederate with a list of trusted domain names of your Auto-Connect™ partners.

Normally, when PingFederate receives an SSO request from a web application at your site (see */sp/startSSO.ping*), the runtime engine completes the connection automatically using metadata obtained from a standard, public location—`http://saml.<domain_name>`. (See *Using Auto-Connect*.) Alternatively, if an Auto-Connect partner elects not to use the standard location, you can supply the applicable URL.

To add a domain:

- Enter a Domain Name and click **Add**.

To specify a URL for metadata retrieval:

1. Click **Advanced View**.
2. Enter the Domain Name if you have not already done so.
3. Enter the Metadata Service URL.

   This entry must be obtained from your Auto-Connect partner.
4. Click **Add**.

   Note: Once you have added the URL, you cannot return to the **Basic View** unless you first remove the URL value using the procedure below.

To edit an entry:

1. Click **Edit** under Action for the entry.
2. Make your change and click **Update**.

To delete an entry:

- Click **Delete** under Action for the entry.

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 50).

The chapter provides instructions for configuring the WS-Trust STS.

- *Server settings* on page 470
- *Identity provider STS configuration* on page 472
- *Service provider STS configuration* on page 483

Server settings

To use the PingFederate WS-Trust STS for partner connections, start by enabling the WS-Trust protocol under Server Settings on the Roles and Protocols screen (see *Choose roles and protocols* on page 161). 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—(see *Configure STS authentication* on page 471).

Enable the WS-Trust protocol

1. Go to the **Server Configuration** > **Server 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 **Server Configuration** > **Server 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

On the **Server Configuration** > **Server 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.

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 *Configure PingFederate properties* on page 98).
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, which involves:

- *Manage token processors* on page 472
- *Manage STS request parameters* on page 475 (Optional)
- *Configure SP connections for STS* on page 476

Manage token processors

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 50.)

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.

- 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: 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 **Server Configuration** > **Server Settings** > **System Options** screen.

For simplicity, this topic focuses on configuring an instance of one of the integrated token processors. For add-on token generators, please refer to the online documentation referenced in the download package.

Select a token processor type

The first step in creating a token-processor instance is choosing the processor type.

- 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

Depending on the selected token processor, the **Instance Configuration** screen presents you with different parameters.

- For token processors bundled with PingFederate, refer to one of the following sections:
  - *Configure a Username Token Processor instance* on page 473
  - *Configure a Kerberos Token Processor instance* on page 473
  - *Configure an OAuth Token Processor instance* on page 474
  - *Configure a JSON Web Token Processor instance* on page 474
  - *Configure a SAML Token Processor instance* on page 474
- For add-on processors, please refer to the online documentation referenced in the download package or the *Knowledge Center*.

Configure a Username Token Processor instance

The integrated Username Token Processor accepts and validates username security tokens.

- 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. Click **Move up** and **Move down** 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

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 510.

- 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 *Configure Active Directory domains or Kerberos realms* on page 219.)

  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 219).

Configure an OAuth Token Processor instance

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 65 and *Access token management* on page 300.)

- 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

The PingFederate STS provides validation for JSON web tokens (JWTs).

- 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

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.

- 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

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.

- 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

On the **Token Attributes** screen, you can choose to mask attribute values that PingFederate logs from this processor instance at runtime.

- 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

1. On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Token Processor Instance** task.
2. On the **Manage Token Processor Instances** screen, click **Save** to retain the configuration of the token processor instance.

   If you want to exit without saving the configuration, click **Cancel**.

Manage STS request parameters

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 478).

You manage request contract on the **Identity Provider** > **STS Request Parameters** screen.

- 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

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).

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

You can configure an STS connection to an SP partner either in conjunction with browser-based SSO or independently.

- 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 **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

Configure protocol settings for IdP STS

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 64 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

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.

- 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

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.

- To continue, click **Configure Token Creation**.

Define an attribute contract for IdP STS

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 *Attribute contracts* on page 79). You identify these attributes on the **Attribute Contract** screen.

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 *Fulfill the attribute contract for token creation* on page 479).

   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 *Fulfill the attribute contract for token creation* on page 479).
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 79.)
3. Click **Add**.
4. Repeat until all applicable attributes are defined.

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

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 475).

- 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

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**.

- 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

On the **Token Processor Instance** screen, choose an instance of a deployed token processor that suits your requirements for this connection.

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 77.

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

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.

- 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 472).

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

When you multiplex one connection for multiple environments (see *Multiple virtual server IDs* on page 92), 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.

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.

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

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.

- 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.

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

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.

- 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 605.

  Repeat to add additional attribute sources.

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.

Fulfill the attribute contract for token creation

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.

- 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 **Server Configuration** > **Server 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 **Custom**

  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 *Attribute mapping expressions* on page 593.
- **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.

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

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 *Multiple virtual server IDs* on page 92), 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 *Issuance criteria and multiple virtual server IDs* on page 596).

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.

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 Custom (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 **Server Configuration** > **Server 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 you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.

   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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the IdP token processor mapping

When you have finished adding a new or modifying an existing token processor instance, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Select a request error handling method

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

When you are finished with the **Token Creation** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review the IdP STS settings

When you are finished with the **WS-Trust STS** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Service provider STS configuration

This section covers the SP configuration for STS, including:.

- *Manage token generators* on page 483
- *Configure IdP connections for STS* on page 485

Manage token generators

The PingFederate Security Token Service (STS) uses token generators to issue security tokens that can be consumed by web services at your site. You must configure at least one generator 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 50.)

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.

- 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: 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 **Server Configuration** > **Server Settings** > **System Options** 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

The first step in creating a token-generator instance is choosing the generator type.

- 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

Depending on the selected token generator, the **Instance Configuration** screen presents you with different parameters.

For the integrated SAML 1.0 and 2.0 Token Generators, refer to the following table and specify parameters for generated SAML tokens.

(For add-on processors, please refer to the online documentation referenced in the download package or the *Knowledge Center*.)

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 **Server Configuration** > **Server 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 198).
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).
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 204).

Extend a token generator contract

Token generators allow administrators to add to a built-in list of user attributes that the generator includes in the outgoing token—an extended generator-attribute contract.

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.

- Enter the name of the desired attribute and click **Add**.

  Repeat this step as needed to add another attribute.

Review the token generator configuration

1. On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Token Generator Instance** task.
2. On the **Manage Token Generator Instances** screen, click **Save** to retain the configuration of the token generator instance.

   If you want to exit without saving the configuration, click **Cancel**.

Configure IdP connections for STS

You can configure an STS connection to an IdP partner either in conjunction with browser-based SSO or independently.

- To configure an STS connection, select the **WS-Trust STS** check box.

  Note: Before this option can be selected, the WS-Trust protocol must be enabled in Server Settings (see *Server settings* on page 470).

Configure protocol settings for SP STS

- 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

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.

- To continue, click **Configure Token Creation**.

Define an attribute contract for SP STS

An attribute contract is the set of user attributes expected in incoming SAML assertions (see *Attribute contracts* on page 79). 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.

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.

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

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**.

- 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

On the **Token Generator Instance** screen, choose an instance of a deployed token processor that suits your requirements for this connection.

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 77.

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

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.

- 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 483).

Restrict a token generator to certain virtual server IDs

When you multiplex one connection for multiple environments (see *Multiple virtual server IDs* on page 92), 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.

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.

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

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.

- 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 605.

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.

Fulfill the attribute contract for token generation

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.

- 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 **Server Configuration** > **Server Settings** > **WS-Trust STS Settings** screen.
  - **LDAP**, **JDBC**, or **Custom**

    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 *Attribute mapping expressions* on page 593.
  - **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.

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

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 *Multiple virtual server IDs* on page 92), 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 *Issuance criteria and multiple virtual server IDs* on page 596).

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.

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 Custom (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 **Server Configuration** > **Server 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 you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.

   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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the SP token generator mapping

When you have finished adding a new or modifying an existing token generator instance, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review the token generation configuration

When you are finished with the **Token Generation** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

Review the SP STS configuration

When you are finished with the **WS-Trust STS** task, you can review the configuration on its **Summary** screen.

- If you need to make any changes, click the heading over the information you want to edit.
- If you are creating a new connection and want to keep your configuration, click **Done**.
- If you are editing an existing configuration and want to keep your changes, click **Save**.

IdP-to-SP bridging

The IdP-to-SP Bridging links on the Server Configuration menu provide access to advanced federation settings.

This chapter covers:

- *Adapter-to-adapter mappings* on page 490
- *Token translator mappings* on page 494
- *Federation hub and authentication policy contracts* on page 90

Note: The information in this chapter is presented from the viewpoint of an administrative user with "Admin" permissions (see *Account management* on page 114).

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).

The mapping may also be used to enable the Google Apps Password Manager (available separately with the Google Apps Connector—see *Outbound provisioning for IdPs* on page 85).

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 loopback 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. For more information, see:

- *Manage IdP adapters* on page 323
- *Manage SP adapters* on page 401

Manage mappings

On the **Adapter-toAdapter Mappings** screen you can add, modify, or delete adapter-to-adapter mappings.

To add a mapping:

- Select the adapter Source Instance (IdP) and Target Instance (SP) and 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:

- Click the mapping name.

To delete a mapping:

- Click **Delete** under Actions for the mapping and then click **Save**.

Assign a license group

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.

To assign a License Group:

- Select the License Group from the drop-down list and click **Next**.

Configure attribute lookup for adapter-to-adapter mapping

Attribute sources are specific data store or directory locations containing information that may be required to fulfill the SP adapter contract.

This portion of the adapter-to-adapter configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

- If data-store lookup is *not* required, click **Next**.
- If data-store lookup is required, click **Add Attribute Source** and complete the setup steps (see *Choose a data store* on page 605).

To modify an attribute source configuration:

1. Click the attribute source Description link.
2. Click **Save** on the screen you change.

   Note: Depending on what you change, you may need to modify dependent data in subsequent steps, as indicated. Click **Save** or **Done** when either of those options appears.

Define adapter-to-adapter contract fulfillment

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 597). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 593. For syntax and examples, see sections under *Construct OGNL expressions* on page 594.)
- LDAP/JDBC/Custom (if a data store is configured)

  Values are returned from a data source. When you make this selection, the Value list is populated by the LDAP, JDBC, or Custom attributes you identified.

  Note: PingFederate appends a description in parentheses for configured data store lookups (see *Configure attribute lookup for adapter-to-adapter mapping* on page 491).
- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 593). 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 605) and attribute is any of the data store attributes you select.

To map attributes:

1. Choose a Source for each SP Adapter Contract attribute.

   See the list under *Map each attribute to fulfill the Adapter Contract from one of these Sources* above.
2. Choose (or enter) a Value for each attribute.

   All values must be mapped.
3. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

Configure a default target URL (optional)

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 405).

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 542).

Define issuance criteria for adapter-to-adapter mapping

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.

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 Custom (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 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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the summary screen

When you have finished configuring Adapter-to-Adapter Mapping, you can review the configuration on the Summary screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the Manage Mappings screen.

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 62). 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 471). 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. For more information, see:

- *Manage token processors* on page 472
- *Manage token generators* on page 483

Manage token mappings

On the **Token Translator Mappings** screen you can add, modify, or delete token-to-token mappings.

To reach this screen:

1. Click **Server Configuration**
2. Click **Token Translator Mappings** under IdP-to-SP Bridging.

To add a mapping:

- Select the token-processor Source Instance and the token-generator Target Instance and 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 542).

To edit a mapping:

- Click the mapping name.

To delete a mapping:

- Click **Delete** under Actions for the mapping and then click **Save**.

Configure attribute lookup for token mapping

Attribute sources are specific data store or directory locations containing information that may be required to fulfill a token-generator contract.

This optional portion of the configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

- If data-store lookup is *not* required, click **Next**.
- If data-store lookup is required, click **Add Attribute Source** and complete the setup steps (see *Choose a data store* on page 605).

To modify an attribute source configuration:

1. Click the attribute source Description link.
2. Click **Save** on the screen you change.

   Note: Depending on what you change, you may need to modify dependent data in subsequent steps, as indicated. Click **Save** or **Done** when either of those options appears.

Define token contract fulfillment

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 597). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 593. For syntax and examples, see sections under *Construct OGNL expressions* on page 594.)

- LDAP/JDBC/Custom (if a data store is configured)

  Values are returned from a data source. When you make this selection, the Value list is populated by the LDAP, JDBC, or Custom attributes you identified.

  Note: PingFederate appends a description in parentheses for configured data store lookups (see *Fulfillment by data store queries* on page 603).
- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 593). 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 603) and `attribute` is any of the data store attributes you select.

To map attributes:

1. Choose a Source for each Token Generator Contract attribute.

   See the list under *Map each attribute to fulfill the Token Generator Contract from one of these Sources:* above.
2. Choose (or enter) a Value for each attribute.

   All values must be mapped.
3. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

Define issuance criteria for token translator mapping

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.

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 Custom (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 **Server Configuration** > **Server 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 you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.

   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.

Related concepts
About token authorization on page 83
Attribute mapping expressions on page 593

Review the token mapping summary screen

When you have finished configuring token-to-token mapping, you can review the configuration on the Summary screen.

If you need to make any changes, click the heading over the information you want to edit. When you are finished, click **Done** and then **Save** on the Manage Mappings screen.

Bundled adapters

PingFederate comes bundled with a set of adapters:

- *HTML Form Adapter* on page 498
- *HTTP Basic Adapter* on page 508
- *Kerberos Adapter* on page 510
- *OpenToken Adapter* on page 513
- *Composite Adapter* on page 519

HTML Form Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or 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 users for credentials.

If *customer IAM* is configured and enabled, the users can optionally register local accounts or sign on using third-party identity providers. If the users 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 users are 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*.

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 the HTML Form Adapter

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** workflow.
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. Click **Move up** and **Move down** 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.

Property	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`.
Session State	How the session state is maintained between adapters of the same type. When **Globally** is selected, sessions are shared among other HTML Form Adapter instances that use the same **Session State** field value ('**Globally**'). When **Per Adapter** is selected, a session is tied to a specific adapter instance. When **None** is selected, a session is not maintained. The default selection is **Per Adapter**.
Session Timeout	The number of idle minutes before a 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. The default value is `60` (minutes).
Session Max Timeout	The maximum lifetime (in minutes) before a session expires regardless of whether the **Session Timeout** field value has been reached. Ignored if **None** is selected for the **Session State** field. 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 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 session does not time out because the user has been actively making SSO requests; however, the 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, sessions do not expire (until PingFederate restarts or the 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, 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.

Property	Description
	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 151.
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 531). **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) 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 *Configure proxied authorization* on page 173). 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.
Change Password Email Notification	When selected, an email notification is sent to 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 an email server. If you have not yet configured the required email server settings in PingFederate, click **Manage Email Server Settings**. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports this email notification feature. This check box is not selected by default.

Property	Description
Show Password Expiring Warning	When selected, the HTML Form Adapter displays a warning to an authenticated user if the password associated with the account is about to expire soon. The message provides the number of days until the expiry of the current password and the options to change the password immediately or to snooze the message. Both the threshold and the snooze interval are configurable in the **Advanced fields** section; the default values are 7 days and 24 hours, respectively. **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.
Password Reset Type	Select one of the following methods for self-service password reset (SSPR). • **Email One-Time Link**: users receive an email with a URL to reset their password. • **Email One-Time Password**: users receive an email with a one-time password (OTP) to reset their password. **Note:** Both email delivery methods require an email server. If you have not yet configured the required email server settings in PingFederate, click **Manage Email Server Settings**. • **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. • **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. This option also requires SMS provider settings (see *the last sub step*). • **None**: users cannot reset password through this HTML Form Adapter instance. The default selection is **None**. When a selection other than **None** is made, as users initiate SSO requests and are prompted to enter their username and password, users have the option to reset their password. As needed, you may also provide your users the Password Reset endpoint shown on the **Summary** screen. The Password Reset endpoint allows users to change their password without submitting SSO requests (see the `/ext/pwdreset/Identify` section */ext/pwdreset/Identify* on page 531). **Note:** To enable password reset, you must also select the **Allow Password Changes** check box. In addition, the LDAP Username PCV is the only PCV bundled with PingFederate that supports self-service password reset.

Property	Description
	If an email server is configured, PingFederate sends an email notification to the user who has successfully reset the password through the HTML Form Adapter. The destination is the user's email address, specifically the value of the attribute defined by the **Mail Attribute** setting in the LDAP Username PCV instance.
Account Unlock	Enables or disables the ability for users to unlock their network 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 Password Reset endpoint shown on the **Summary** screen. The Password Reset endpoint allows users to unlock their account without submitting SSO requests (see the `/ext/pwdreset/Identify` section */ext/pwdreset/Identify* on page 531). **Note:** You must also select a **Password Reset Type** type 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 self-service 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. 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.
Enable Username Recovery	Enables or disables the ability for users to recover their username via email 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 Password Reset endpoint shown on the **Summary** screen. The Password Reset endpoint allows users to recover their username without submitting SSO requests (see the `/ext/pwdreset/Identify` section */ext/pwdreset/Identify* on page 531). **Note:** This capability requires an email server. If you have not yet configured the required email server settings in PingFederate, click **Manage Email Server Settings**. 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 sends an email message containing the recovered username. The destination is the email address provided by the user. This check box is not selected by default.

e) Click **Show Advanced Fields** to review or modify default values.

For information about each field, see *HTML Form Adapter advanced fields* on page 505.

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 389). 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 395).
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.)
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 URL is `<PingFederate base URL>/ext/<Logout Path>`. 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.
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	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 **Enable 'Remember My Username'** check box 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. 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.
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`.
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 sends username recovery email messages to 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.

CAPTCHA options

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.

HTTP Basic Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or 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 users 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 the HTTP Basic Adapter

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** workflow.
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. Click **Move up** and **Move down** 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.

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.
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**.

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 the Kerberos Adapter

1. If you have not already done so, log on to the PingFederate administrative console.
2. Go to the **Identity Provider** > **Adapters** screen.
3. On the **Manage IdP Adapter Instances** screen, click **Create New Instance**.
4. On the **Type** screen, enter an **Instance Name** and **Instance ID**, select **Kerberos Adapter** from the list, and then click **Next**.

   The **Instance ID** may not contain spaces or underscores.
5. 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.
6. Click **Next**.

   Note: If this is a child instance, select the override check box related to the settings you want to modify.
7. On the **IdP Adapter** screen, select the **Domain/Realm Name** for your Windows domain. If the domain or realm you want does not appear, click **Manage Active Directory Domains/Kerberos Realms** to add it (see *Configure Active Directory domains or Kerberos realms* on page 219).
8. Optional: 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 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.
9. Optional: Click **Show Advanced Fields** and make any desired changes for the following settings.

Field	Descriptions
Error Template (optional)	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	Descriptions
Authentication Context Value (optional)	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.)

10. Click **Next**.
11. On the **Adapter Attributes** screen, select **Username** (and optionally **Domain/Realm Name**) to be used in constructing a unique identifier (pseudonym) for account linking at the SP.

    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.

    You can also choose to mask the values of any or all attributes that PingFederate logs from the adapter at runtime.

    If OGNL expressions might be used to map derived values into outgoing assertions and you want those values masked, select the related check box under the Attribute list.
12. Click **Next**.
13. On the **Summary** screen, click **Done**.
14. On the **Manage IdP Adapter Instances** screen, click **Save**.

    Important: You must click **Save** if you want to retain the adapter configuration.

Configure end-user browsers

Follow the steps in the following sections to configure client-side browsers at your site in order to use the Kerberos Adapter to authenticate users:

- *Configure Microsoft Internet Explorer* on page 512
- *Configure Mozilla Firefox* on page 513

Note: The client-side configuration requires the **Base URL** value of your PingFederate environment, which can be found in **Server Configuration** > **Server Settings** > **Federation Info**.

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

To configure Internet Explorer (9 or higher) for Kerberos authentication, review the following settings in Internet Options.

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

To configure Firefox for Kerberos authentication, configure Firefox as follows:

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.

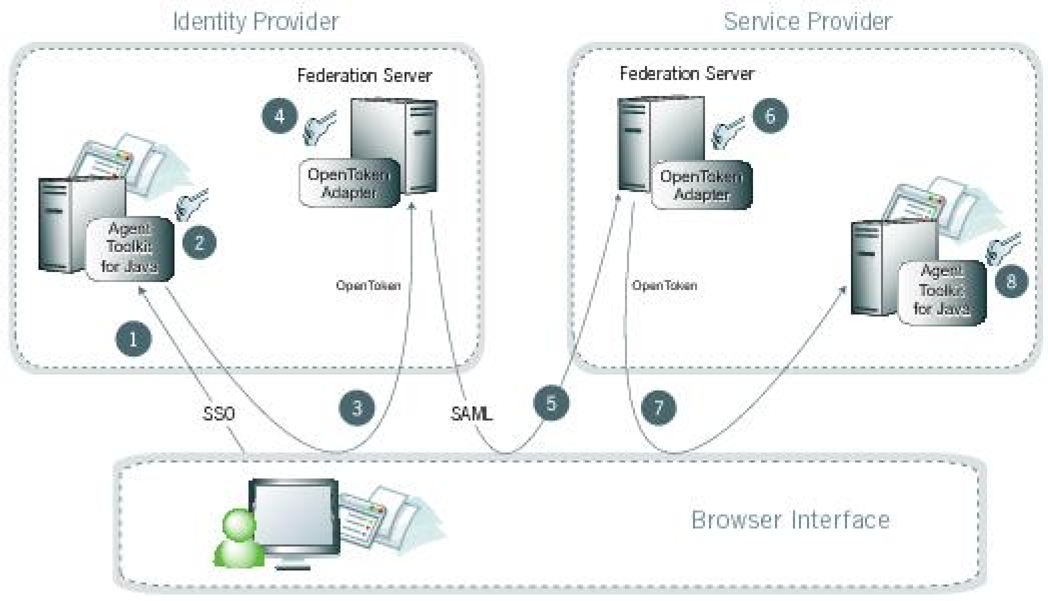


Figure 24: IdP-Initiated SSO: POST/POST

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 the OpenToken IdP Adapter

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** workflow.
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. Refer to the on-screen field descriptions and the following table for more information.

   Note: These values are dependent on your developer's implementation.

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.	
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, preceded by a period (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.

Field	Description
Use Verbose Error Messages	If selected, use verbose TokenException messages.
Obfuscate Password	If selected (the default), the password is obfuscated and password-strength validation is applied. Clearing the check box allows backward compatibility with previous OpenToken agents.
Session Cookie	If selected, OpenToken is set as a session cookie (rather than a persistent cookie). Applies only if the **Transport Mode** field is set as **Cookie**. The check box is not selected by default.
Secure Cookie	If selected, the OpenToken cookie is set only if the request is on a secure channel (https). Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Delete Cookie	If selected, the token cookie is deleted immediately after consumption. Applies only if the **Transport Mode** field is set to **Cookie**. The check box is not selected by default.
Replay Prevention	Selecting this option is recommended only if **Query Parameter** is the chosen token transport mode and form POST is used by an associated connection to send the SAML assertion. If selected, PingFederate ensures that the token can be used only once. 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.

   Note that the OpenToken IdP Adapter always extends the core contract with an attribute `userId` 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 the OpenToken SP Adapter

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. Refer to the on-screen field descriptions and the following table for more information.

   Note: These values are dependent on your developer's implementation.

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.	
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**.
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.

Field	Description
Cookie Domain	The server domain, preceded by a period (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.

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 the Composite Adapter

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** workflow.
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 (as described in the following table), and then click **Update**.

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.

Column	Description
	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**.

Self-service user account management

PingFederate provides various self-service applications for end users to manage their network 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.

For configuration steps, refer to the following topics:

- *Configure self-service password management* on page 522
- *Configure self-service account recovery* on page 523
- *Configure self-service username recovery* on page 526

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 379 and *Enable profile management* on page 394.

Configure self-service password management

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 send email notifications when users have successfully changed the password associated with their accounts through the HTML Form Adapter or when the 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 email notifications for change password and password expiry are not supported at this point.

1. On the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen, configure an email server if you want PingFederate to send email notifications in the event that a user has successfully changed the password associated with the account through the HTML Form Adapter, the network password is about to expire, or both.
2. On the **Server Configuration** > **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 *Configure proxied authorization* on page 173).

   This step does not apply if you are validating credentials through the PingOne® Directory PCV.
3. On the **Server Configuration** > **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 instance.

   If you are validating credentials through the LDAP Username PCV and if you want to enable email notifications, follow this *sub step* 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 email 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 used for personalizing messages to the user. The default value is `displayName`.
Mail Attribute	The LDAP attribute containing the email address, to which PingFederate sends email notifications to the requesting users. The default value is `mail`.

4. 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, follow from this *sub step* 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 Email Notification** check box if you want PingFederate to send an email notification to 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.

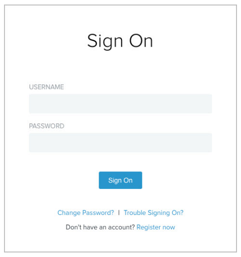
e) Click **Show Advanced Fields** to review or modify default values related to the change password capability.

   For example, update the **Change Password Template** field if you want to use a custom template to render the **Change Password** screen.

5. Optional: Customize and localize the on-screen messages and email messages.

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

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 four different mechanisms:

- 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 self-service password reset, 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.

1. On the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen, configure an email server if you want to enable password reset via email, notify the users about the status of their request (for all password reset mechanisms), or both.
2. On the **Server Configuration** > **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 *Configure proxied authorization* on page 173).

3. On the **Server Configuration** > **Password Credential Validators** screen, create a new LDAP Username PCV instance.

   You can also reuse an existing LDAP Username PCV instance. If so, follow this *sub step* 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 password reset.

      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 used for personalizing messages to the user. The default value is `displayName`.
Mail Attribute	The LDAP attribute containing the email address, to which PingFederate sends email notifications to the requesting users. 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. There is no default value.

4. 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, follow from this *sub step* to configure your adapter instance to enable the self-service password reset 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 Email Notification** check box if you want PingFederate to send an email notification to 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.

Property	Description
Password Reset Type	Select one of the following methods for self-service password reset. • **Email One-Time Link**: users receive an email with a URL to reset their password. • **Email One-Time Password**: users receive an email with a one-time password (OTP) to reset their password. • **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. • **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. This option also requires SMS provider settings. • **None**: users cannot reset password through this HTML Form Adapter instance. This is the default selection.

f) Select the **Account Unlock** check box if you want to enable self-service account unlock as well.
g) Click **Show Advanced Fields** to review or modify default values related to self-service password reset.

For self-service password reset using PingID, you must download the settings file from the PingOne® admin portal and upload the file to the **PingID Properties** advanced field. In addition, if you have not enabled PingFederate integration in your PingID account, you must do so as well.

h) 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.

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).
From Number	The sender number in the text message notifications.

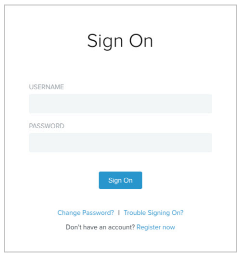
Tip: For additional information about each field or Twilio, please refer to *support.twilio.com*.

Once saved, these SMS provider settings apply to all services using text message notifications.

5. Optional: Customize and localize the on-screen messages, email messages or text message.

You have now successfully created a new instance or modified an existing instance of the HTML Form Adapter with the self-service password reset 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 Password Reset 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

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.

1. On the **Server Configuration** > **Server Settings** > **Runtime Notifications** screen, configure an email server that PingFederate can use to send username recovery email messages.
2. On the **Server Configuration** > **Data Stores** screen, create a new LDAP data store.

   You can also reuse an existing LDAP data store connection.
3. On the **Server Configuration** > **Password Credential Validators** screen, create a new LDAP Username PCV instance.

   You can also reuse an existing LDAP Username PCV instance. If so, follow this *sub step* 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 used for personalizing messages to the users. 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.

Field	Description
	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 send username recovery email messages to users who have proven ownership of their email addresses (see this *sub step*). There is no default value.

4. On the **Identity Provider** > **Adapters** screen, create a new HTML Form Adapter instance.

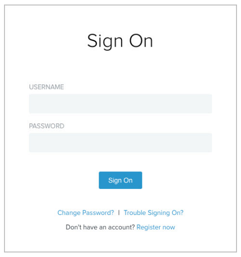
   You can also reuse an existing HTML Form Adapter instance. If so, follow from this *sub step* to configure your adapter instance to enable the self-service username recovery capability.

   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) 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.
5. Optional: Customize and localize the on-screen messages and email messages.

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 Password Reset 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 132).

SP endpoints also include those available for SCIM inbound provisioning (see *Provisioning for SPs* on page 85).

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 490) and retrieval of WS-Trust metadata (see *WSC and WSP support* on page 63).

PingFederate provides a favorite icon for all Application Endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 154.

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* on page 567.

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 350). 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 222).
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 *Configure the HTML Form Adapter* on page 499).
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 336) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 162).

/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 321). 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.

Parameter	Description
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 352).

/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 verification email messages 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 verification email messages, 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 **Server Configuration** > **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 187.)

/ext/pwdreset/Identify

The Password Reset 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 **Server Configuration** > **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 187.)

Related concepts
Customizable user-facing screens on page 132

Related tasks
Localize messages for end users on page 147

SP endpoints

The following sections describe the PingFederate SP endpoints, including the query parameters that each accepts or requires:

- *SP services* on page 532
- *SCIM inbound provisioning endpoints* on page 536

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* on page 567.

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.

Parameter	Description
	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 405) or for individual connections (see *Configure default target URLs* on page 431), 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 132).

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 425).

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. 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.
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	The index number of your site's Assertion Consumer Service, where you want the assertion to be sent.

Parameter	Description
(optional - SAML 2.0)	
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 *Define SP affiliations* on page 376).
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 413) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 162).
Domain	The domain name associated with the requesting user's IdP to invoke Auto-Connect™. In this case, PartnerIdpId cannot be used (see *Auto-Connect* on page 94).

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 402).

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 352).

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 132).

/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 *IdP Discovery*). This endpoint is similar to `/sp/startSSO.ping` and accepts the same parameters, with the exception of PartnerIdpId and vsid (see */sp/startSSO.ping* on page 532). 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*).

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.

Parameter	Description
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.
SharedSecret	Used to authenticate the initiating application. The AppId and SharedSecret must both match the application authentication settings within the PingFederate server. **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 SP* on page 336) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 162).

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.)

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

HTTP method	Description
	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.
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.

HTTP method	**Description**
	The group ID is set as the id attribute in the JSON response, and the full URL to reference the group is in the HTTP response Location header.
	For an existing group, you can also use the POST method to either update or delete the group by appending the group ID to the path (in the format of `/pf-scim/v1/Groups/`*`group_id`*) and setting the request header X-HTTP-Method-Override value to `PUT` or `DELETE`, respectively. (For more information, see the PUT and DELETE method descriptions at the end of this topic.)
GET	`/pf-scim/v1/Groups`
	• Retrieves all group records.
	• A successful response is indicated by an HTTP 200 status code and a list of all groups and their attributes.
	`/pf-scim/v1/Groups/`*`group_id`*
	• Retrieve the group record of a specific group.
	• A successful response is indicated by an HTTP 200 status code and the requested group record.
	`/pf-scim/v1/Groups?attributes=`*`attribute`*
	• Retrieves the specific attribute from all groups.
	• A successful response is indicated by an HTTP 200 status code and a list of the desired attribute from all groups.
	Note: For more information, see *3.2.2 List/Query Resources* in SCIM Specification (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.

HTTP method	Description
	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.
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 153).

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 *Configure runtime reporting* on page 156).

/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 490).

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 132).
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 402).
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 *Configure the HTML Form Adapter* on page 499).

/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 494).

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 *Configure PingFederate properties* on page 98).

/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 **Server Configuration** > **Server 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* on page 544.

/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.

Parameter	Description
(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 **Server Configuration** > **Server 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* on page 544.

Construct an alternative metadata exchange endpoint

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).

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 also endpoints for the clients to revoke tokens, the clients to retrieve OpenID Connect metadata (for example, the supported ID token signing algorithms), the resource owners to revoke authorization grants, and developers to submit client registrations using the OAuth 2.0 Dynamic Client Registration protocol.

The topic describes OAuth-developer information on PingFederate endpoints for the OAuth AS. Unless otherwise indicated, these endpoints and associated parameters are defined in the aforementioned specifications.

Note: Append each endpoint to the base URL of your PingFederate environment; for example:

- https://www.example.com:9031/as/authorization.oauth2
- https://www.example.com:9031/.well-known/openid-configuration

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 authorization. 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.

Endpoint: /as/authorization.oauth2

The table below shows parameters for this endpoint:

Parameter	Description
client_id (Required)	The client identifier.
response_mode	When set to `form_post`, the authorization response is returned to the client in an auto-POST form in accordance with *the OAuth 2.0 Form Post Response Mode specification* (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	Applicable only when response_type parameter value is `code`. Supply a one-time string value used to associate the authorization request with the token request to reduce the risk of code interception attack. For more information, refer to *Proof Key for Code Exchange (PKCE) by OAuth Public Clients* (tools.ietf.org/html/rfc7636). **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 552).
code_challenge_method	Applicable only when the response_type parameter value is `code` and a code_challenge parameter value is provided.

Parameter	Description
	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	Required if more than one redirection URIs are configured in PingFederate for the client or if a wildcard is used for a single URI entry. Optional for clients with only one specific redirection URI configured. Note that 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 552).
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 using the key obtained from the pre-configured JWKS URL or the JWKS. 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 authorization 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 using the key obtained from the pre-configured JWKS URL or the JWKS. 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 279 (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).
scope	The scope of the access request expressed as a list of space-separated, case-sensitive strings. Valid scope values are defined in the **OAuth Server** > **Authorization Server Settings** screen. **Tip:** Scopes can be restricted per client.

Parameter	Description
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 table below displays 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.
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`

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).

Endpoint: /as/token.oauth2

Like other OAuth 2.0 endpoints, append the token endpoint to the **Base URL** value defined on the **Server Configuration** > **Server Settings** > **Federation Info** screen. If the **Token Endpoint Base URL** field is configured on the **OAuth Server** > **Authorization Settings** screen, the token endpoint is available at both locations. For instance, if the **Base URL** and the **Token Endpoint Base URL** fields are set to `https://localhost:9031` and `https://www.example.com:9031`, respectively, then the token endpoints are available at:

- https://localhost:9031/as/token.oauth2, and
- https://www.example.com:9031/as/token.oauth2

Note: Per OAuth specifications, this endpoint accepts only the HTTP POST method.

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 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/

Authentication method	Parameters
	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: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 *Proof Key for Code Exchange (PKCE) by OAuth Public Clients* (tools.ietf.org/html/rfc7636), see *Authorization endpoint* on page 545.
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 545). The value here must match the authorization-request value, if applicable.

Parameter	Description
	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.

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 by the resource owner. If omitted, the scope is treated as equal to that originally granted by the resource owner. Valid scope values are defined in the **OAuth Server** > **Authorization Server Settings** screen. Scopes can also be restricted per client.

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.
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 `<pf_install>/pingfederate/server/default/conf/language-packs/pingfederate-messages.properties` file.

As needed, they may be localized using the PingFederate localization framework for an international audience.

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

These parameters apply when the grant_type parameter for `/as/token.oauth2` is set to `client_credentials`.

Parameter	Description
scope	The scope of the access request.

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.

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.

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

These parameters apply 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. 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 (applicable only to access tokens using the reference-token data model).

Example

```
{
  "scope":"read edit admin",
  "token_type":"urn:pingidentity.com:oauth2:validated_token",
  "expires_in":3172,
  "client_id":"super_cool_mobile_client",
  "access_token":
  {
    "uid":"sfHqhad9onMjXsQNI1mZP9mD7AQasmskd",
    "group":["employee","sales","manager"],
    "email":"user@example.com"
  }
}
```

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 *Configure an OAuth client* on page 279).

Once an ATM instance is chosen, PingFederate takes into consideration the per-instance session validation settings and processes the validation request (see *Manage session validation settings* on page 305).

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.

Endpoint: /as/introspect.oauth2

Note: Per OAuth specifications, this endpoint accepts only the HTTP POST method.

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 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/

Authentication method	Parameters
	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 *Configure an OAuth client* on page 279).

Once an ATM instance is chosen, PingFederate takes into consideration the per-instance session validation settings and processes the validation request (see *Manage session validation settings* on page 305).

Introspection parameters

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:

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.
- 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.

If a token is invalid, the OAuth AS returns `{"active":false}` to the client.

Example 1: A response for a valid access token

```
{
  "sub":"joe",
  "Username":"joe",
  "active":true,
  "OrgName":"Ping Identity Corporation",
  "token_type":"Bearer",
  "exp":1467008819,
  "client_id":"ac_client",
  "username":"joe"}
```

Example 2: A response for a valid refresh token

```
{"active":true}
```

Example 3: A response for an invalid token

```
{"active":false}
```

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.

Endpoint: /as/revoke_token.oauth2

Note: Per OAuth specifications, this endpoint accepts only the HTTP POST method.

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.

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 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.

Authentication method	Parameters

	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.

Parameters

The token revocation endpoint uses the following parameters using the `application/x-www-form-urlencoded` format in the HTTP request entity-body:

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`

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.

Grant-management endpoint

The grants endpoint (two are provided, one for use with parameters) is where end-users/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.

Endpoint: /as/clients.oauth2

Note: Per OAuth specifications, this endpoint accepts only the HTTP POST method.

Both the request and the response follow RFC7591. Here are some examples.

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.eyJzY29wZSI6WyJkQ1IiXSwiY2xpZW50X2lkX25hbWUiOi
CHtcQ79Wefz2Sw5GOB5LfV9mWJ0n3vzJ93Ie7wbEAkalIFg53J-9e7s59MjA1igx6ybflGMQ9QAjYobs-
jM24arJZZgopEXvcx6IQpyU8U4AMTJ7tr9Lmody8P0QZOKcUDBTT5egv9vr5NuXCtUBfVPhGZ-3p5g5mwrr
oDhmilbmiga4319YSFfX5-
U3li9XPeN3JZB2ukLbTFjjVIVLJIInbSR_IFTWP5Irg92aXLrIfm5MvBp8D1fOU6xYjbgjvw9QKNiFFVD7c
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"
}
```

OpenID Provider configuration endpoint

The OpenID Provider configuration endpoint provides configuration information for the OAuth clients to interface with PingFederate using the OpenID Connect protocol.

Note: This endpoint is only active when the OAuth AS role and the OpenID Connect protocol are enabled on the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

The configuration information returned by this endpoint is contolled by a template file and can be customized to suit multiple use cases simultaneously.

Endpoint: /.well-known/openid-configuration

This public endpoint accepts HTTP GET requests without authentication and the following query parameter.

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.

Example (truncated)

```
$ curl -s https://localhost:9031/.well-known/openid-configuration|python -
m json.tool
{
  ...
  "issuer": "https://localhost:9031",
  "authorization_endpoint": "https://localhost:9031/as/
authorization.oauth2",
  "token_endpoint": "https://www.example.com:9031/as/token.oauth2",
  "revocation_endpoint": "https://localhost:9031/as/revoke_token.oauth2",
  "userinfo_endpoint": "https://localhost:9031/idp/userinfo.openid",
  "introspection_endpoint": "https://localhost:9031/as/introspect.oauth2",
  "jwks_uri": "https://localhost:9031/pf/JWKS",
  "registration_endpoint": "https://localhost:9031/as/clients.oauth2"
  "ping_revoked_sris_endpoint": "https://localhost:9031/pf-ws/rest/
sessionMgmt/revokedSris",
    "ping_end_session_endpoint": "https://localhost:9031/idp/startSLO.ping",
    ...
    "scopes_supported": [
      "openid",
      "address",
      "email",
      "phone",
      "profile",
      ...
    ],
    ...
    "claims_supported": [
      "address",
      "birthday",
      ...
    ],
    ...
    "response_types_supported": [
      "code",
      "token",
      "id_token",
      ...
    ],
    ...
    "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"
    ],
    ...
    "id_token_signing_alg_values_supported": [
      "none",
      "HS256",
      "HS384",
      "HS512",
      "RS256",
      "RS384",
      "RS512",
      "ES256",
      "ES384",
      "ES512",
      "PS256",
      "PS384",
      "PS512"
    ],
    ...
    "token_endpoint_auth_signing_alg_values_supported": [
      "RS256",
      "RS384",
      "RS512",
      "ES256",
      "ES384",
      "ES512",
      "PS256",
      "PS384",
      "PS512"
    ],
    ...
    "request_object_signing_alg_values_supported": [
      "RS256",
      "RS384",
      "RS512",
      "ES256",
      "ES384",
      "ES512",
      "PS256",
      "PS384",
      "PS512"
    ],    ...
  }
```

Some notable metadata parameters

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://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 265 and *Token endpoint* on page 549).

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

Digital signature algorithms

The id_token_signing_alg_values_supported, token_endpoint_auth_signing_alg_values_supported, and request_object_signing_alg_values_supported parameters provide lists of supported algorithms to process digital signatures.

In this example, because PingFederate is integrated with a hardware security module (HSM) and configured to use static keys for OAuth and OpenID Connect, the endpoint includes additional RSASSA-PSS digital signature algorithms (`PS256`, `PS384`, and `PS512`) in its response. (For more information on HSM integration and static keys, see *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, respectively.)

Related information
OpenID Connect Discovery 1.0 (openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata)

Related tasks
Configure OpenID Connect policies on page 315
Customize the OP configuration endpoint response on page 153

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 **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

Endpoint: /idp/userinfo.openid

This endpoint accepts HTTP GET requests without parameter. Clients must present valid access tokens.

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: The self-contained access token in the Authorization HTTP header is truncated for readability only.

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 308); 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

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 — 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 126).

- 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 *Authentication* on page 210.

PingFederate also provides the following REST-based web services:

- *OAuth Client Management Service* on page 574 — An API at the runtime port to manage OAuth client applications.
- *OAuth Access Grant Management Service* on page 583 — Allows retrieval and revocation of access grants.
- *Session Revocation API endpoint* on page 586 — Allows OpenID Connect clients to query revocation status of their sessions and add end-user sessions to the revocation list.
- *PingFederate administrative API* on page 588 — An 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 331 or *Access IdP connections* on page 410.

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 331 or *Access IdP connections* on page 410). 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 572).
- `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 *Authentication* on page 210). 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/id` 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/id

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).

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.
requireSignedRequests	When set to true, the client is required to send its authorization requests 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.

Parameter	Description
	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)
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). • 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 *Configure PingFederate properties* on page 98). • 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 **Server Configuration** > **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`.
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 or to transmit request parameters

Parameter	**Description**
	in signed request objects (or to do both) 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 authorization requests and without the use of signed request objects for some other transactions. (For runtime processing, see *Authorization endpoint* on page 545.) If the client signs its JWTs using an RSASSA-PSS signing algorithm, PingFederate must be integrated with a hardware security module (HSM) to process the digital signatures. (For more information on HSM integration, see *Supported hardware security modules* on page 56.)
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 *Assertion Replay Prevention Service* on page 644).
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.
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; the authorization consent page is not presented to the user for this client. A valid value is either `true` or `false`. If not provided, a value of `false` is assumed.
restrictScopes	Set to `true` to restrict this client to access specific common scopes or scope groups as specified by the restrictedScopes parameter. If the restrictedScopes parameter is not provided or provided without any scope or scope group, only the default scope is available for the client. A valid value is either `true` or `false`. If not provided, a value of `false` is assumed.
restrictedScopes	Used in conjunction with the restrictScopes parameter value of `true` to restrict this client to access specific common scopes or scope groups in addition to the default scope.
exclusiveScopes	A list of exclusive scopes or scope groups available for this client. If not provided, no exclusive scope or scope group is allowed for this client.
grantTypes	An array of one or more grant types allowed for the client. Allowed values: • `authorization_code` • `password` (Resource Owner Password Credentials)

Parameter	**Description**
	• `refresh_token` (Use with `authorization_code` or `password`) • `client_credentials` • `implicit` • `extension` (JWT Bearer Token or SAML 2.0 Bearer Assertion) • `urn:pingidentity.com:oauth2:grant_type:validate_bearer` (Access token validation)
restrictedResponseTypes	An array of one or more response_type parameter values that this client can use. 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`

Parameter	**Description**
defaultAccessTokenManage	The default Access Token Management 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

Parameter	Description
	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.
persistentGrantExpirationType	Overrides the **Persistent Grant 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, regardless of the global setting. • `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`.
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`.
refreshRolling	Indicates whether a new refresh token is issued with each new access token. A valid value is either `true` or `false`. When not provided, the **Roll Refresh Token Values** setting configured globally setting in the **OAuth Server** > **Authorization Server Settings** screen is used.

extendedParameters	Provide values for extended client metadata fields; for example:
	
	`{`
	`  ...,`
	`  "extendedParams": {`
	`    "entry": [`
	`      {`
	`        "key": "ContactName",`
	`        "value": {`
	`          "elements": "J. Smith"`
	`        }`

Parameter	Description

```
      },
      {
        "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 **OAuth Server** > **Client Settings** > **Extended Client Metadata** screen.)

Note: The rest of the parameters are only applicable when the **OpenID Connect** protocol is enabled in **Server Configuration** > **Server Settings** > **Roles & Protocols** screen and this client supports the OpenID Connect use cases.

Parameter	Description
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 an integration with a hardware security module (HSM) and a static-key configuration for OAuth and OpenID Connect. (For more information on HSM integration and static keys, see *Supported hardware security modules* on page 56 and *Manage keys for OAuth and OpenID Connect* on page 205, 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.
policyGroupId	The desired Open ID Connect policy.

Parameter	Description
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 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 for 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 305.)

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 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)..

Sample JSON

```
{
  "client": [
    {
      "secret":
 "L1u508MfeZYTvR03kcpa6ezysNEspFEtzxSAIEOTl18AuNd2pnNqjkRdOXzfTFXc",
      "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.

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 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":
 "L1u508MfeZYTvR03kcpa6ezysNEspFEtzxSAIEOTl18AuNd2pnNqjkRdOXzfTFXc",
      "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/id

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.

Related concepts
PingFederate administrative API on page 588
Grant types on page 66
Access token management on page 300
Back-Channel Session Revocation on page 321
Asynchronous Front-Channel Logout on page 321

Related tasks
Define an OAuth client data store on page 183
Configure an OAuth client on page 279
Configure AS settings on page 265
Define scopes on page 268

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 daa store.

Manage by user (based on the USER_KEY value)

```
/pf-ws/rest/oauth/clients/<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 *Persistent vs. transient grants* on page 69.)

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.

Related tasks
Manage Password Credential Validator instances on page 211
Configure AS settings on page 265
Define an OAuth client data store on page 183
Configure an OAuth client on page 279

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 client secrets via HTTP Basic authentication or client certificates.

Note: Generally speaking, if clients are allowed to add sessions to the revocation list, consider enabling the **Check session revocation status** option for 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 305.)

Endpoint: /pf-ws/rest/sessionMgmt/revokedSris

This resource accepts the *POST* and *GET* methods. It also requires the X-XSRF-HEADER HTTP header with any value to prevent cross site request forgery.

Note: The POST method described in this section requires the element name/value pair formatted in 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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT2
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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT2
X-XSRF-HEADER: PingFederate
```

If PingFederate has been configured to manage authentication sessions in the **Identity Provider (or Service Provider)** > **Sessions** screen, 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.

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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT2
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
- OAuth settings
- Server settings

For a complete list, see *Using the API interactive documentation* on page 592. For known limitations, see *Release notes*.

Authentication

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 is protected by:

- *Native authentication* — Uses credentials managed within PingFederate.
- Alternative authentication — Utilize credentials external to PingFederate through an LDAP user-data store, the RADIUS protocol, or client certificates:
  - *LDAP authentication*
  - *RADIUS authentication*
  - *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 older), 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 native 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 **Server Configuration** > **Configuration Archive** screen in the administrative console
- The `/pf-mgmt-ws/ws/ConnectionMigrationMgr` and `/pf-mgmt-ws/ws/ConfigReplication` Connection Management Web Service endpoint
- The **Server Configuration** > **Application Authentication** screen in the administrative console for the **Connection Management** service

Enable native authentication

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.

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 **Account Management** 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 **Account Management** 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 118.
3. On the **Server Configuration** > **Account Management** 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 589.

   Note: When assigning role(s), keep in mind that all users defined in the **Account Management** screen can be used to access the administrative API and the administrative console.

Enable LDAP authentication

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.

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 589.

   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

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.

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 589.)

   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

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 192).

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.

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 589.

   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.

Using the API interactive documentation

PingFederate ships with interactive documentation for both developers and non-developers to explore the API endpoints, view documentation for the API, and experiment with API calls. 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.

To access the interactive documentation in PingFederate

1. Start PingFederate.
2. Start a web browser.
3. 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 using the interactive documentation:

1. Click a section of the administrative API you would like to explore—for example, **/sp/idpConnections**.
2. Expand the method you want to use; for example, **GET**.
3. Enter any required parameters.
4. Click **Try it out**.

   The **Request URL**, **Response Body**, **Response Code**, and **Response Headers** appear.

   Note: You may be prompted to log in using administrative credentials over HTTP Basic Authentication. Enter the credentials of any existing administrator. The role of the administrator affects their access to the requested API.

Attribute mapping expressions

PingFederate provides an advanced option allowing administrators to map user attributes by way of an expression language. Because the option carries with it a potential for misuse, however, it is disabled in the administrative console for security reasons.

Tip: When importing a configuration archive that uses expression mapping, the feature is enabled automatically.

This topic describes the option, which is based on the Object-Graph Navigation Language (OGNL), and how to enable or disable it.

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

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.

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 **Server Configuration** > **Cluster Management** screen, click **Replicate Configuration**.
   c) Restart PingFederate on the engine nodes.

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 “Contractor” or “Employee” group.

```
#this.get("ds.ldap.memberOf")!=null?
(
  #this.get("ds.ldap.memberOf").toString().matches("(?i)
   .*CN=Contractor,cn=users,dc=company,dc=com.*|
   .*CN=Employee,cn=users,dc=company,dc=com.*")
):@java.lang.Boolean@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::74d
```

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.oau
#claims.get("aud")
```

Retrieving the entire private key JWT:

```
#this.get("context.HttpRequest").getObjectValue().getParameter("client_assertion")
```

Using the OGNL edit screen

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.

- 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.

  Here is an example of the edit screen, from the IdP configuration flow:
- 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

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.

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

Message Types	Available Variables Classes/Interfaces in Javadoc
	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
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	see below

```
<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

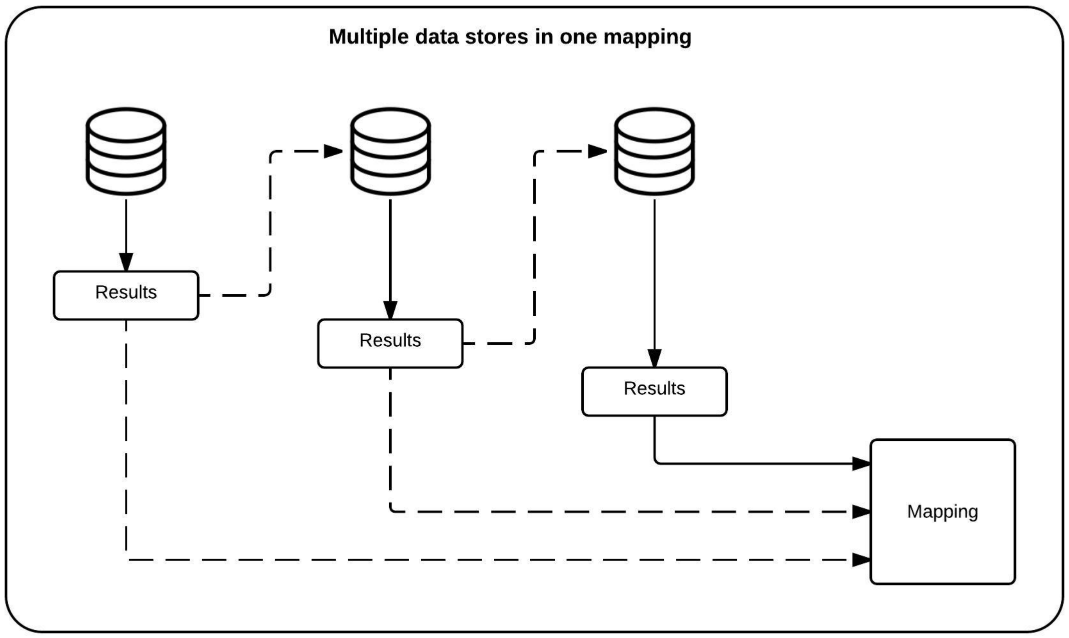
PingFederate can be configured to use data stores to supply attributes for various contracts (for example, adapter contracts, browser-based SSO token contract, or STS token contracts) or conditions to be verified in the token authorization process. PingFederate supports a wide varierty of directory servers and database servers for user-attribute lookup. Alternatively, you can create your own driver for non-JDBC or non-LDAP data stores, such as flat files or SOAP-connected databases, using the PingFederate Custom Source SDK, and then map from custom fields to fulfill your use cases.

Attribute mapping with multiple data sources

When querying data stores for attributes to help fulfill a contract on the IdP side, PingFederate can be configured to use more than one attribute source.

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*.

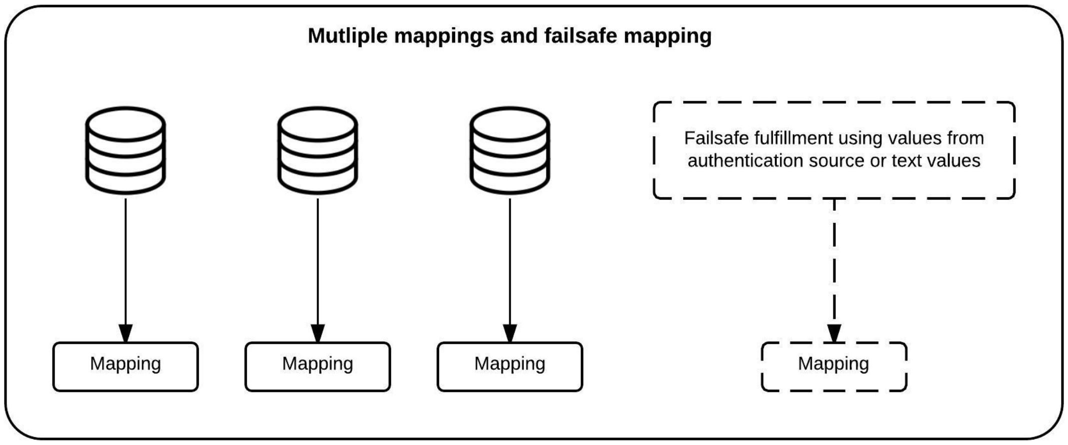
If a required attribute (such as SAML_SUBJECT in a SAML contact or subject in an authentication policy contract) cannot be fulfilled, the request fails.

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:

- OpenID Connect policies (the user-attributes mapping process)
- Resource-owner credential mappings
- IdP adapter mappings
- Access token mappings

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

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

On the **Data Store** screen, choose a data store for PingFederate to look up attributes.

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	• *Specify a JDBC database table and columns* on page 605 • *Enter a database search filter* on page 606
LDAP	• *Specify an LDAP directory and attributes* on page 607 • *Define encoding for LDAP binary attributes* on page 608 (optional) • *Enter an LDAP search filter* on page 608
Custom	• *Specify a custom source filter and fields* on page 609

Specify a JDBC database table and columns

When you choose to use a database source for attributes, 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 JDBC 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 JDBC 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 JDBC 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 connection string of your JDBC 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*.

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

On the **Database Filter** screen, enter a JDBC `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 from your own database, using any other columns from the targeted table.

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 a JDBC data store.

   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 an LDAP directory and attributes

When you choose to use an LDAP source for attributes, 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 LDAP root.
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.

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 LDAP 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 LDAP binary attributes

The **LDAP Binary Attribute Encoding Types** screen appears when a binary attribute is added on the **LDAP Directory Search** screen. (Attributes are specified as binary in the **Server Configuration** > **Data Stores** > ***the applicable LDAP connection*** > **LDAP Configuration** > **Advanced** screen.) Since binary attribute data cannot be used in an assertion to the SP, specify the encoding type (`Base64`, `Hex`, or SID) that you want to apply during fulfillment.

Note: Defining encoding for LDAP binary attributes is only applicable to IdP and IdP-to-SP bridging use cases.

To set an encoding type, select a value from the **Attribute Encoding Type** list.

Repeat this step for each LDAP attribute that has been specified as binary in the LDAP connection.

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 an LDAP search filter

On the **LDAP Filter** screen, enter a LDAP 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 LDAP 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 from your own LDAP server, using any 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. An LDAP 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.

1. Enter an LDAP search filter in the text field.
2. Ensure the syntax and variable names are correct. For general information about search filters, consult your LDAP documentation.
3. Click **Next** to complete the configuration to query attributes from an LDAP data store.

   Later in the workflow, you can use the attribute values returned from your LDAP 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 LDAP 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 a custom source filter and fields

When you choose to use a custom source for attributes, you follow this path through the configuration steps.

On the **Configure Custom Source Filter** screen you begin to specify a filter, or search query, for your custom data source. This screen display and the syntax of the filter depends on your developer's implementation of the custom source SDK.

On the **Configure Custom 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.

Later in the workflow, you can use the values returned from the custom source in the applicable contract fulfillment screen, the issuance criteria screen, or both.

Configure failsafe options

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 chose to retrieve attributes from multiple data stores using a single mapping 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.

- 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

Review your data store configuration on the **Summary** screen.

- To restart the configuration process, click **Cancel**.
- To make modifications, click the heading over the information you want to edit, and then proceed with your changes.
- To keep your configuration, click **Done** to continue with the rest of the configuration, where you can use the attribute or column values returned from the data store in subsequent sibling configuration items, such as contract fulfillment or token authorization, to fulfill your use case.

  If you are modifying an existing configuration, click **Save** to commit your changes.

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

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.

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, particularially 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.

Updating the log level for existing `Logger` elements does not require a restart. The adjustments are activated within half a minute. Other changes require a restart of PingFederate; for example:

- Adding new `Logger` elements
- Updating the size attribute in the `RollingFile/Policies/SizeBasedTriggeringPolicy` element
- Updating the max attribute in the `RollingFile/DefaultRolloverStrategy` element
- Updating the filePattern attribute without modifying the fileName attribute in the `RollingFile` appender

Resolve startup issues

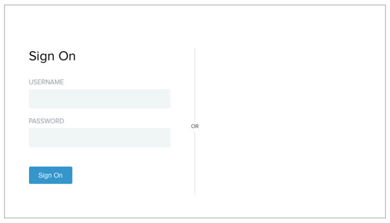
Problem	Solution
PingFederate does not start.	Make sure that Oracle Java SE Runtime Environment (Server JRE) is installed (see *Install Java* on page 19).
The server starts but indicates the license file is not found or invalid.	Ensure a current license is installed (see *Manage PingFederate license* on page 96).

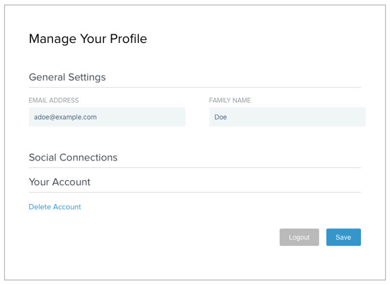
Troubleshoot data store issues

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 *Configure a JDBC database connection* on page 169.
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 *Configure an LDAP connection* on page 171). 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 192).

Troubleshoot registration and profile management issues

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.

- Verify whether users are using browser extensions that may have caused this issue. Disabling or fine-tuning the browser extension should resolve the issue.

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** in 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 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 for PingFederate 7.2 (or higher).
- 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

You can configure PingFederate 7.2 (or higher) 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:

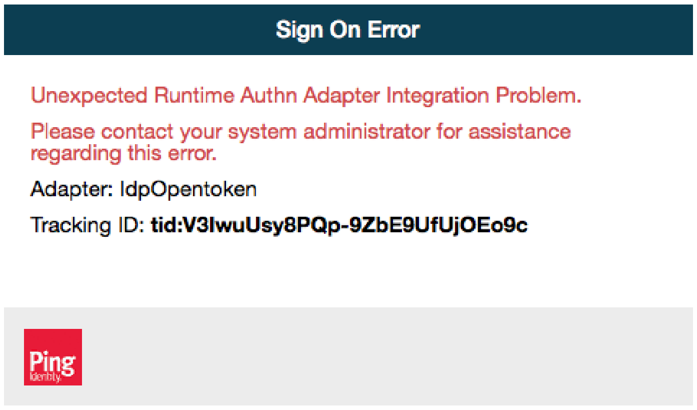
- `general.error.page.template.html`
- `generic.error.msg.page.template.html`
- `idp.slo.error.page.template.html`
- `idp.sso.error.page.template.html`
- `sourceid-wsfed-idp-exception-template.html`
- `sp.slo.error.page.template.html`
- `sp.sso.error.page.template.html`
- `state.not.found.error.page.template.html`

Note: You can find these Velocity template files in the `<pf_install>/pingfederate/server/default/conf/template` directory.

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.

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 tracking ID

All server log messages (except the contents of the inbound requests and the outbound responses) are prefixed with their respective tracking IDs, which helps locating related log messages and payloads for a given transaction for troubleshooting.

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.

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.

Correlate log messages by PF cookie

If you are using PingFederate 7.1 (or older), 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.

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.

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
```

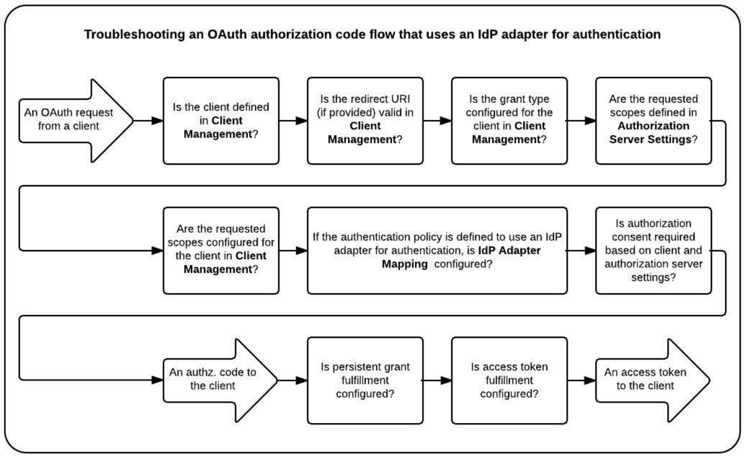
After reviewing the related log messages based on the tracking ID (`V3IwuUsy8PQp-9ZbE9UfUjOEo9c`), you found the next few messages:

```
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.

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

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.

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 end 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 end user is prompted with a list of all available authentication sources. The end 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 end user an authorization consent page 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) Go to the **OAuth Server** > **Client Management** screen.
d) Select the applicable client.
e) 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 end 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:

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.

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 *Configure runtime notifications* on page 155).
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

Access token

A data object by which a client authenticates to a resource server and lays claim to authorizations for accessing particular resources.

Account link

A persistent name identifier that enables federation of separately established accounts among disparate domains (see also *account linking* and *pseudonym*).

Account linking

A form of identity mapping among separate user accounts managed under different domains. The mapping typically involves a name identifier, which may be a pseudonym, to link the user to each account. The identifier is persisted at the SP site to enable seamless SSO/SLO. Additional attributes may be sent with the identifier.

Account mapping

A form of identity mapping by which one or more user attributes is passed in a single sign-on transaction. The attributes are used at the destination site as a means identifying the user and looking up local account information.

Adapter

Plug-in software that allows PingFederate to interact with web applications and authentication systems (see *SSO integration kits and adapters* on page 72).

Adapter contract

A list of attributes “hard-wired” to an adapter and conveyed generally via cookies between the adapter and application.

Artifact

A reference to a SAML protocol message. The federation partner that receives the artifact dereferences it, identifying the sender, and requests the complete message in a separate SOAP transaction.

Artifact Resolution Service

The SOAP endpoint that processes artifacts returned from a federation partner to retrieve the referenced XML message. Can be used to dereference authentication requests, assertion responses, and SLO messages.

Assertion

A SAML XML document that contains identifying information about a particular subject; for example, a person, company, application, or system. A SAML assertion can contain authentication, authorization, and attribute information about the subject.

Assertion Consumer Service

A SAML-compliant portion of PingFederate in an SP role that receives and processes assertions from an IdP.

Attributes

Distinct characteristics that describe a subject. If the subject is a website user, attributes may include a name, group affiliation, email address, and attributes alike.

Attribute contract

A list of attributes, agreed to by the partners in an identity federation, representing information about a user (SAML subject). The attributes are sent from the IdP to the SP during SSO or STS processing.

Attribute mapping

A form of identity mapping between IdP and SP user accounts that uses attributes to identify the user or provide supplemental information.

Attribute source

An data source used to fulfill a requestor's attribute contract.

Audience

The XML element in a SAML assertion that uniquely identifies a Service Provider.

Authentication context

An element in a SAML assertion indicating the method or process used by an IdP to authenticate the subject of the assertion; may be used for authorization decisions or auditing compliance.

Authentication request (OpenID Connect)

An OAuth 2.0 authorization request using extension parameters and scopes defined in the OpenID Connect specifications that a relying party (RP, an OAuth client) sends to an OpenID Provider (an OAuth authorization server) for the purpose of authenticating the end user.

Authentication request (SAML 2.0)

A SAML XML document that a service provider (SP) sends to an identity provider (IdP) to request that the IdP to authenticate the identity of an end user and to return a response for the request.

<AuthnRequest>

See *Authentication request (SAML 2.0)*

Authorization request

A request based on the *OAuth 2.0 Authorization Framework* (tools.ietf.org/html/rfc6749) that an OAuth client sends to an authorization server for the purpose of obtaining an access token (for the purpose of ultimately accessing protected resources on a resource server).

Attribute source

Specific database or directory location containing data needed by an IdP to fulfill a connection partner's attribute contract or by an SP to look up additional attributes to fulfill an adapter contract.

Back-channel

Server-to-server, cross-domain communication path using a protocol, typically SOAP, that does not rely on a browser as an intermediary.

Binding

A mapping of SAML request and response messages to specific transport protocols (redirect, POST, or artifact).

Certificate

A digital file used for identity verification and other security purposes. The certificate, which is often issued by a certificate authority (CA), contains a public key, which can be used to verify the originator's identity.

Certificate Revocation List (CRL)

A list of revoked signing certificates, maintained by the issuing authority at a public URL.

Channel

A dedicated outbound provisioning configuration specific to a particular service partner, data source, and target service.

Connection partner

Entities, such as companies, that are part of an identity federation. These entities are referred to as connection partners in the PingFederate configuration process.

Credential

Information used to identify a subject for access purposes (e.g., username and password). A credential can also be a certificate.

Database management system

A system for storing and maintaining user account information and attributes. The tables and columns in the RDBMS are used by PingFederate to create user look-up and attribute retrieval queries. (See *Java Database Connectivity*.)

Data store

A database or directory location containing user account records and associated user attributes.

Data Encryption Standard (DES)

A symmetric-key method of encryption.

Defederation

Optional user-initiated delinking of an identity federation that uses a persistent name identifier or pseudonym for account linking.

Digital signature

A process for verifying the identity of the originator of an electronic document and whether the document has been intercepted or altered. The process involves message signing, signature validation, and signing policy coordination between partners.

Endpoint

A terminal or gateway that generates or terminates a stream of information. For example, a PingFederate SP server contains an endpoint for the Assertion Consumer URL.

Entity ID

The XML element in a SAML assertion that uniquely identifies an identity provider.

Extensible markup language

A structured, hierarchical text format, based on SGML (Standard Generalized Markup Language), for the flexible and organized exchange of data.

Grant type

The intermediate credentials that represent a resource owner authorization. Grant types are exchanged by the client with the OAuth authorization server in order to obtain an access token.

HTTP cookie

Information sent from a server to a web browser to identify a registered website user. Once the cookie is placed in the browser, it is sent back to the server to identify the user every time the user accesses the site. PingFederate's integration adapters interface with the cookie.

HTTP header

The section of an HTTP request or response containing information about the client or the server. PingFederate can use HTTP headers to look up session information passed by the IdP's web application.

HTTP request parameter

A named parameter sent as part of a URL request from a browser to a web server.

ID token

A security token that contains claims (attributes) about the authentication of an end user (represented by the sub claim in the ID token) by an authorization server when using an OAuth client, and potentially other requested claims. The ID token is represented as a JSON Web Token (JWT).

Identity federation

A trust agreement between or among organizations, implemented using accepted standards, to provide user-authentication tokens and other user or system attributes securely across domains, primarily to enable cross-domain SSO.

Identity provider (IdP)

The identity source or SAML authority that authenticates a subject and provides an SP with a security assertion vouching for that authentication.

IdP-initiated SSO or SLO

An identity federation transaction in which the initial action requiring a security context from an IdP occurs at a IdP's site. For example, the user is logged on to the IdP and requests protected resources on an SP. The IdP sends authentication information to the SP.

Inbound

A direction of message flow coming into a server relative to the server's identity federation role (IdP or SP). For an IdP, inbound messages include SAML authentication requests. For an SP, inbound messages include SAML assertions.

Java database connectivity (JDBC)

A Java API that allows Java programs to interact with databases.

Kerberos ticket

The security token for the Kerberos protocol.

Key Distribution Center

The control center for authentication and authorization for Kerberos.

Keysize

The length (in bits) of each key in a keypair.

Keypair

The private key and public key represented by a certificate. PingFederate uses the private key of its keypair(s) to generate signatures for assertions, requests, and responses, as applicable.

Lightweight Directory Access Protocol (LDAP)

A set of protocols used for accessing information directories. PingFederate uses the LDAP v3 protocol for user look-up and attribute processing.

Metadata

The SAML 2.0 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.

Network access server (NAS)

A RADIUS client server that provides a single point of access to a protected resource.

OAuth

A standard framework that enables an application (OAuth client) to obtain access tokens from an OAuth authorization server for the purpose of retrieving protected resources on a resource server.

OAuth authorization server (AS)

A server that issues access tokens to clients (sometimes on behalf of a resource owner) for use in authenticating a subsequent representational state transfer (REST) API call.

OAuth client

An application that desires access to a resource protected by a resource server and interacts with an authorization server to obtain access tokens to do so.

OIDC

See *OpenID Connect*.

Certificate Status Protocol (OCSP)

(OCSP) A standard developed by the Internet Engineering Task Force that enables applications to obtain the current status of signing certificates, indicating whether a certificate has been revoked, via HTTP.

Opaque

Not readable. If a user's subject identifier is opaque, an SSO partner cannot directly identify the user with reference to the source. An persistent identifier, or *pseudonym*, can be used for Account Linking.

OpenID Connect

OpenID Connect 1.0 is an identity layer on top of the OAuth 2.0 protocol. This protocol enables a relying party (OAuth client) to verify the identity and to obtain claims (attributes) about an end user from an authorization server (an OpenID Provider).

OpenID Provider (OP)

An OAuth 2.0 authorization server that is capable of authenticating end users and providing claims on their behalf to relying parties.

Outbound

A direction of message flow leaving a server. For an IdP, outbound messages include SAML assertions. For an SP, outbound messages include SAML authentication requests.

Partner

See *connection partner*.

Portal

A Web-based application, accessed using a web browser, that often aggregates content from multiple providers, serves as a central point of entry, or both.

POST

An HTTP method of transmitting data contained in HTML forms, by which the data appears in the message body.

Primary domain controller (PDC)

A role that is assigned to a particular server participating in a Windows network.

Principal

A user, system, or process whose identity can be authenticated. See *subject*.

Profiles

Rules that describe how to embed SAML assertions into and extract them out of other protocols in order to enable SSO or SLO. Profiles describe SAML request and response flows that fulfill specific use cases.

Protected resource

Information, typically accessed via a web URL, that is protected by an access management system. See *target URL*.

Protocol

An agreed-upon format for transmitting data. XML format of SAML request or response messages.

Pseudonym

A persistent name identifier assigned to a user and shared among entities, usually with the user's permission, to enable SSO and SLO. Pseudonyms are often used with the SAML account linking protocol to enable SSO while preventing the discovery of the user's identity or activities.

Public key infrastructure (PKI)

Enables users of an unsecured public network, such as the Internet, to securely and privately exchange data and money through the use of keypairs and certificates. The PKI provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates.

Redirect

A SAML binding that conveys a request or response by sending the user's browser to another location. For instance, an authentication request can be sent from an SP through a browser to an IdP.

Refresh token

A long-lived token used by the client to obtain a new access token without having to obtain fresh authorization from the resource owner.

Relying party (RP)

An OAuth 2.0 Client that requires end-user's authenticity and claims (attributes) from an OpenID Provider.

Remote Authentication Dial-in User Service (RADIUS)

A networking protocol for user-access management that includes specifications for two-factor authentication.

<RequestSecurityToken> (RST)

WS-Trust or WS-Federation XML element identifying a request for validation of a security token, or for validation and then issuance of a replacement security token.

<RequestSecurityTokenResponse> (RSTR)

WS-Trust or WS-Federation XML element identifying a response to an RST and containing either the status of the submitted security token or both the status and (if requested and the received token is valid) a newly issued token for further SSO or web-services processing.

Resource server

A server capable of accepting and responding to resource requests on which an access token is presented.

SAML

See *Security Assertion Markup Language*.

SAML authority

A security domain that issues SAML assertions.

Scope

Permissions (for example, creating an event on a calendar) associated with an access token.

Secure sockets layer (SSL)

An encryption protocol that sends data between a client and server over a secure HTTP connection.

Security Assertion Markup Language (SAML)

A standard, XML-based, message-exchange framework enabling the secure transmittal of authentication tokens and other user attributes across domains.

System for Cross-domain Identity Management (SCIM)

A REST-based protocol for provisioning and managing user identities across the Internet (see *www.simplecloud.info*).

Security domain

An application or group of applications that trust a common security token used for authentication, authorization, or session management. The token is issued to a user after the user has authenticated to the security domain.

Security token

A collection of information used to establish acceptable identity for security purposes. Tokens can be in binary or XML format. A SAML assertion is one kind of security token.

Security Token Service (STS)

An entity responsible for responding to WS-Trust requests for validation and issuance of security tokens used for SSO authentication to Web Services.

Service-oriented architecture

A loosely coupled application architecture in which all functions or services are accessible via standard protocols. Interfaces are platform and programming-language independent.

Service provider (SP)

A system entity that provides access to a protected resource based on authentication information supplied by an IdP.

SP-initiated SSO or SLO

An identity-federation transaction in which the initial action requiring a security context from an IdP occurs at a SP's site.

Session persistence

A mechanism for identifying a user or browser for subsequent requests to a server, needed because the HTTP protocol is stateless. This information is used to lookup state information for the user; for example, items in a shopping cart. PingFederate does not implement session persistence; it facilitates the communication of session information between systems that do.

Simple Object Access Protocol (SOAP)

Defines the use of XML and HTTP to access services, objects, and servers in a platform-independent manner.

Single logout (SLO)

The process of logging a user out of multiple “session participants” or sites where the user has started an SSO session.

Single Logout Return Service

The SAML implementation endpoint URL that returns logout requests.

Single Logout Service

The SAML implementation endpoint URL that receives logout requests for processing.

Single sign-on (SSO)

The process of authenticating an identity (signing on) at one website (usually with a user ID and password) and then accessing resources secured by other domains without re-authenticating.

Single Sign-on Service

The SAML implementation endpoint URL that receives authentication requests for processing.

Source ID

A 20-byte sequence used to determine an IdP's identity.

Subject

A person, computer system, or application. In the SAML context, assertions make statements about subjects. See *principal*.

Target URL

The SP's protected resource; the end destination of an SSO event. See *protected resource*.

Transient name identifier

A temporary ID used to preserve user anonymity while facilitating account linking.

Token authorization

A mechanism for evaluating attribute criteria available during a transaction to determine whether a user is authorized to access resources. A token in this instance can mean any type of security token; for example, SSO, session cookie, or OAuth token.

Token exchange

The process by which a security token is exchanged for another security token.

Token translators

An aggregate term for both token processors (used by the IdP PingFederate Security Token Service (STS) to handle different types of incoming security tokens) and token generators (used by the SP PingFederate STS to issue various types of tokens).

Uniform resource identifier (URI)

Identifies a web resource with a string of characters conforming to a specified format.

Uniform resource locator (URL)

Identifies a resource according to its Internet location.

Virtual server ID

An optional unique identifier by which an identity federation deployment can be known to a specific connection partner.

Web Services Security (WSS)

A standard mechanism for securing web service interactions, often by binding a security token to the web service request.

Web services

Non browser-based, loosely coupled applications that provide modular, programming-language-independent access to specific functions and data across the Internet, via XML and standard protocols.

Web service client (WSC)

An entity that requests a Web Service interaction. In the context of an STS, the Web Service Client would request that a security token be issued for the interaction.

Web service enhancement

Supplemental software for the .NET framework provided by Microsoft.

Web service provider (WSP)

In the context of an STS, an entity that requests validation of the security token sent with a client's request for service.

WS-SX

The OASIS committee working on WS-Trust.

WS-Trust

A standard protocol by which an application can request that an STS issue, validate, or exchange security tokens.

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
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
IdM	Identity Management
IdP	Identity Provider
IP	Internet Protocol
J2SDK	Java 2 Software Development Kit
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	
RSTR	

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
SSTC	Security Services Technical Committee (of OASIS)
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 634).

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 *Directed clustering* on page 637).

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. 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 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.

Table 2: Configuration file and service implementation

Configuration file	**RPC-based service implementation**
cluster-account-locking.conf	*Account Locking Service* on page 646
cluster-artifact.conf	*Artifact-Message Persistence and Retrieval Service* on page 644
cluster-assertion-replay-prevention.conf	*Assertion Replay Prevention Service* on page 644
cluster-idp-session-registry.conf	*IdP Session Registry Service* on page 643
cluster-inter-request-state.conf	*Inter-Request State-Management (IRSM) Service* on page 642
cluster-session-revocation.conf	*Back-Channel Session Revocation Service* on page 646
cluster-sp-session-registry.conf	*SP Session Registry Service* on page 643

Table 3: 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	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.

Property	Description
(found only in the `cluster-session-revocation.conf` file)	When set to `true`, queries for revocation status are processed locally. When `false`, the processing node pulls revocation status from other engine nodes in the cluster (subject to the **rpc.timeout** value). **Note:** When adding a session to the revocation list, the processing node always propagates the information to all engine nodes in the cluster (see *Back-Channel Session Revocation Service* on page 646). 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.
- Assertion Reply 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

- 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 **Server Configuration** > **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 table below indicates the configuration file that applies to each implementation. Refer to the indicated sections in this topic for detailed information about each implementation.

Configuration file	RPC-based service implementation
cluster-account-locking.conf	*Account Locking Service* on page 646
cluster-artifact.conf	*Artifact-Message Persistence and Retrieval Service* on page 644
cluster-assertion-replay-prevention.conf	*Assertion Replay Prevention Service* on page 644
cluster-idp-session-registry.conf	*IdP Session Registry Service* on page 643
cluster-inter-request-state.conf	*Inter-Request State-Management (IRSM) Service* on page 642
cluster-session-revocation.conf	*Back-Channel Session Revocation Service* on page 646
cluster-sp-session-registry.conf	*SP Session Registry Service* on page 643

The following table describes the properties contained in the configuration files (the Artifact-Message Persistence and Retrieval Service uses only the rpc.timeout setting):

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.

Property	Description
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.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 recipients 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 *Back-Channel Session Revocation Service* on page 646). 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:

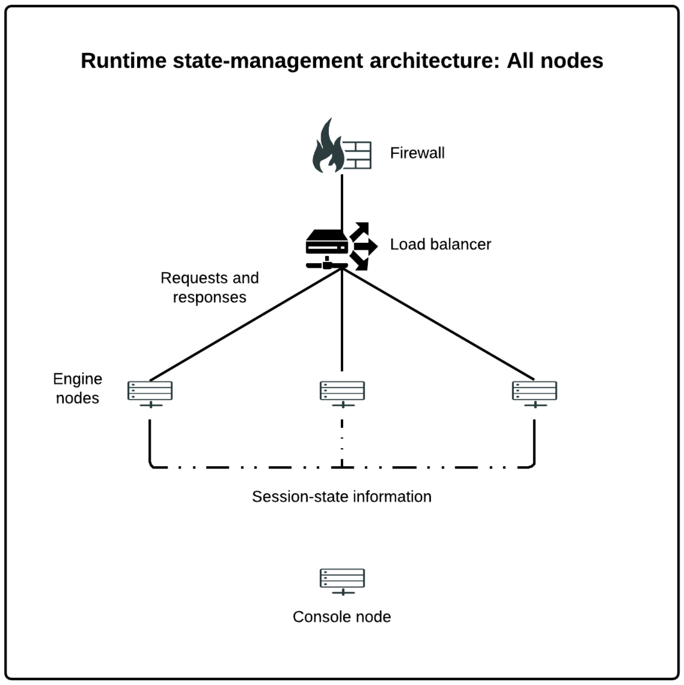
- *Sharing all nodes* on page 639
- *Designating state servers* on page 639
- *Defining subclusters* on page 640

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.

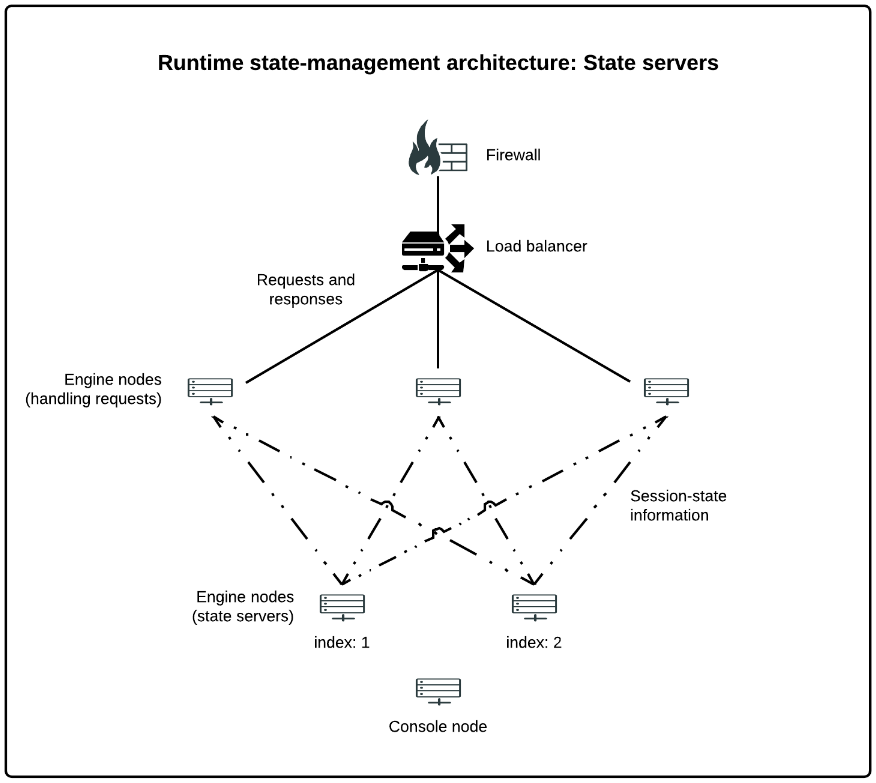

Runtime state-management architecture: All nodes
Firewall
Load balancer
Requests and responses
Engine nodes
Session-state information
Console node

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

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 nodes despite any state server or subcluster setup.

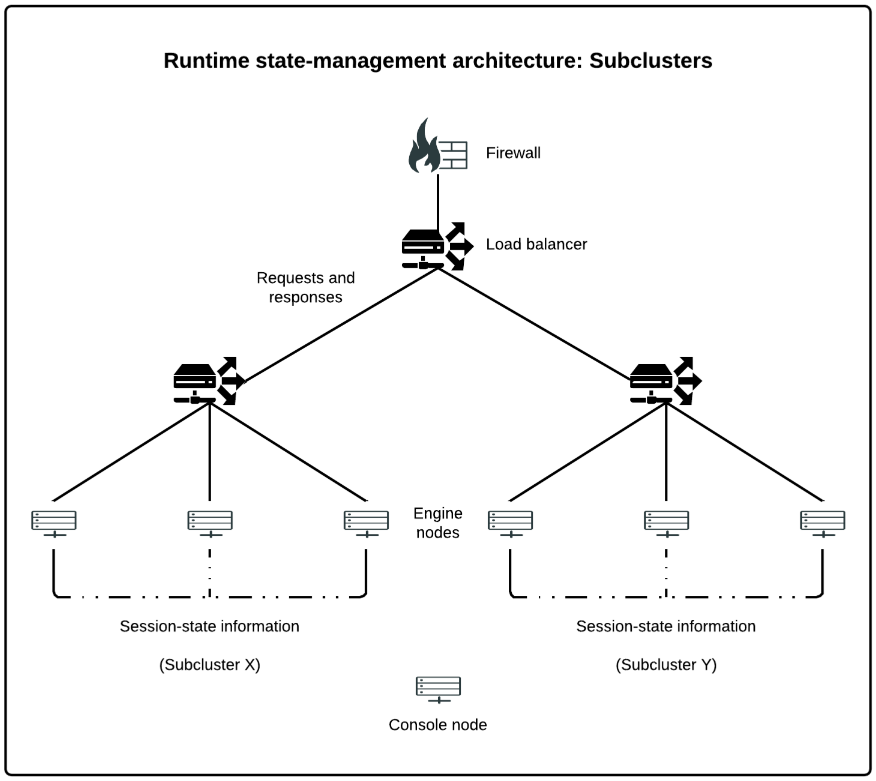
Defining subclusters

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 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` interface chooses from among three methods available to track this state: group RPC-based (the clustering default), cookie based, and local memory-based.

Caution: Depending on deployments, if cookies are used for state management in a cluster instead of the default RPC-based method, keep in mind that the cookies may become large enough to exceed certain browser limitations.

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 configuration file is `cluster-inter-request-state.conf` in the `<pf_install>/pingfederate/server/default/conf` directory.

The service proxy `InterRequestStateMgmtProxy` in the `hivemodule.xml` file is set to use this implementation as the clustering default. The specific class name is:

```
org.sourceid.saml20.service.impl.grouprpc.InterRequestStateMgmtGroupRpcImpl
```

Cookie-based tracking

In a clustered mode, this implementation tracks short-lived session-state data with the help of browser cookies. The implementation works by compressing and encrypting the session-state data and sending it to the user's browser as a cookie. User data is then returned to the server on subsequent requests from the browser. No data for this service needs to be shared among nodes, and even the simplest round-robin load-balancing technique works. There is some overhead in the encryption and compression, but the advantage is that the session state must travel over the network only between the entities that actually need it; the user and the server that handles that user's next request.

Caution: Cookie-based tracking, which can result in larger cookies, should not be used for configurations using account linking. (For more information, see *About account linking*.)

This implementation does not use group RPC to share session-state data. However, RPC is used by default to handle dynamic distribution of the AES keys used for encryption and decryption (unless a static key is configured). The `key-tracker.xml` file, in the `<pf_install>/pingfederate/server/default/data/config-store` directory, controls the interval at which new keys are issued, the maximum number of keys to retain, and the key length. The oldest member of a cluster group is responsible for issuing and distributing new keys.

The file `inter-req-cookie-config.xml`, also in the `config-store` directory, specifies the name of the cookie, which you may modify as needed. You can also configure a static AES encryption key by specifying its hex value (the key must be set on each node). While a static key is somewhat less secure, it eliminates the need for key management between nodes via back-channel communication.

To use this service implementation, set the class attribute for the `InterRequestStateMgmt` service in the `hivemodule.xml` file to the class name:

```
org.sourceid.saml20.service.impl.cookie.InterReqStateMgmtCookieImpl
```

Important: Adaptive clustering does not support this implementation. Use the group RPC-based session tracking instead.

Local memory-based tracking

In this alternative, the inter-request state of a user is tracked in the local memory of the processing server.

Note: To implement this alternative, the load balancer must support sticky sessions to force all requests for the same user session to be routed to the same server.

To use this service implementation, set the class attribute for the `InterRequestStateMgmt` service in the `hivemodule.xml` file to the class name:

```
org.sourceid.saml20.service.impl.localmemory.InterReqStateMgmtMapImpl
```

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 *Directed clustering* on page 637).

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 *Directed clustering* on page 637).

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

When the artifact binding is used to send messages to partners, the PingFederate server must keep track of the message referenced by the artifact until the partner makes a SOAP call to retrieve it.

Two options are available for using outbound artifacts in a cluster:

- Group RPC-based retrieval
- Using SAML 2.0 indexing

Group RPC-based retrieval

When PingFederate is in clustered mode, the service proxy selects a group RPC-based implementation, which takes advantage of node indexing but does not use the preferred-nodes concept. Sticky-session load-balancing strategies are not effective for the artifact binding, because the requests that result in the issuance of the artifact and the artifact resolution request come from different locations.

This implementation works by having the node that issues an artifact encode its node index into the artifact using two bytes of the artifact message handle. When a server receives the artifact resolution request, it uses the encoded node index to determine the issuing node and makes an RPC call to get the associated message.

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.

The service proxy uses the class:

```
org.sourceid.saml20.service.impl.grouprpc.ArtifactPersistenceSvcGroupRpcEncodedNodeIdxImp
```

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.

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 account locking-state information with a replica set. If region identifiers are defined, PingFederate shares account locking-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 account locking-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

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*.

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 647).
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 634 and *Runtime state-management services* on page 641).
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.

   (If you need to request a new license or replace the current one, see *Manage PingFederate license* on page 96.)
8. On the **Server Configuration** > **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.

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

Follow these steps to configure the cluster protocol properties for all PingFederate nodes.

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.)
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.

Property	Description
	(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.
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.

Property	Description
	(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`.)
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. (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.

Tip: Other administrative-console and runtime behavior configuration options are also maintained in the `run.properties` file. For more information, see *Configure PingFederate properties* on page 98.

Enable dynamic discovery for clustering

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. The elastic scaling capability helps you to bring additional PingFederate engine nodes online with no additional configuration changes after the initial setup.

When this discovery mechanism is enabled, a new group member pulls cluster membership information from a centralized repository. If a cluster exists, it joins the group; otherwise, it forms a new group. 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, giving you the flexibility between public and private cloud storage for the dynamic discovery mechanism.

Dynamic discovery requires only a one-time setup. Its configuration is maintained in the `tcp.xml` file located in the `<pf_install>/pingfederate/server/default/conf` directory. Once configured, you are no longer required to coordinate the effort of updating IP addresses on all nodes.

Important: You must manually configure or synchronize the dynamic discovery properties in the `tcp.xml` file on each new 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.

1. Configure all required pf.cluster.* properties in the `run.properties` file in the `<pf_install>/pingfederate/bin` directory.

   Note: Ensure the pf.cluster.transport.protocol property is set to `tcp` (the default value).

2. Edit the `tcp.xml` file in the `<pf_install>/pingfederate/server/default/conf` directory.

   Follow the inline comments to enable dynamic discovery.

Field	Description
IAM roles for Amazon EC2 (`com.pingidentity.aws.AWS_PING`)	
port_number	The port, on which PingFederate listens for cluster communication. The default value is `${pf.cluster.bind.port}`, which pulls the value defined in the `<pf_install>/pingfederate/bin/run.properties` file.
tags	A comma separated list of EC2 tags. When specified, only Amazon Machine Images (AMIs) that have been assigned with the specified tag (or tags) are eligible to join the cluster. If multiple tags are specified, only AMIs that have been assigned with all tags will be considered. For information about tags, see the *documentation* from Amazon EC2 (docs.aws.amazon.com/AWSEC2/latest/UserGuide/Using_Tags.html).
filters	A semi-colon separated list of key value pairs of system metadata. When specified, only AMIs that match the specified filter (or filters) are eligible to join the cluster. If multiple filters are specified, only AMIs that match all filters will be considered. 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 AMIs that have an instance type of either t2.small or t2.medium and also an architecture of x86_64 will be considered.

Field	Description
	For information about filters, see the *documentation* from Amazon EC2 (docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html). You can use a combination of tags and filters, in which case only AMIs that satisfy both criteria are eligible to join the cluster.
access_key and secret_key	The access key and its secret key for the purpose of querying EC2 for AMIs. Keys can be encrypted using the `obfuscate` utility (`obfuscate.bat` for Windows or `obfuscate.sh` for Unix or Linux), located in the `<pf_install>/pingfederate/bin` directory. **Note:** If access_key and secret_key are defined, tags and filters are ignored.
log_aws_error_messages	When set to `true` (the default), error messages received from Amazon EC2 are logged to the server log.
Amazon S3 (`S3_PING`)	
location	The name of the bucket in your Amazon S3 environment. For information about buckets, see the *documentation* from Amazon Web Services (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 Web Services (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 Unix or 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*.)
OpenStack Swift (`SWIFT_PING`)	
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 Unix or 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` fields, 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).

After the initial setup, your nodes are ready to be deployed, undeployed, and redeployed as traffic volume changes.

Note: The configuration and process of the discovery mechanism are independent from those of the runtime state-management services.

The goal of the discovery mechanism is to find a node to retrieve cluster information and to subsequently joins and rejoins such cluster while that of the runtime state-management services are about sharing session-state information with the applicable nodes based on the associated configurations.

Some runtime state-management services support the concept of adaptive clustering, designated state servers, an subcluster; however, 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 634.

Deploy provisioning failover

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.

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`.)
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

Property	Description
	Synchronization Frequency set in the **Server Configuration** > **Server 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 **Server Configuration** > **Server Settings** > **Outbound Provisioning** screen.

   Once configured, if the provisioning servers belong to the same PingFederate clustered environment, go to the **Server Configuration** > **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 running in cluster mode, the administrative console provides a **Server Configuration** > **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, administrators must also use the **Replicate Configuration** screen to initiate the transmission of the license file from the console node to all server nodes.

The replication process takes only a moment, during which time any new requests coming into any of the cluster nodes are held in a queue. When the replication is complete, processing continues where it left off.

Administrators may also click **Refresh Table** to verify that any new nodes have joined the cluster.

Configuration-archive deployment

After you configure or reconfigure the console, you can also update cluster nodes by downloading a configuration archive from the **Server Configuration** > **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	RPC-based service implementation
cluster-account-locking.conf	*Account Locking Service* on page 646
cluster-artifact.conf	*Artifact-Message Persistence and Retrieval Service* on page 644
cluster-assertion-replay-prevention.conf	*Assertion Replay Prevention Service* on page 644
cluster-idp-session-registry.conf	*IdP Session Registry Service* on page 643
cluster-inter-request-state.conf	*Inter-Request State-Management (IRSM) Service* on page 642
cluster-session-revocation.conf	*Back-Channel Session Revocation Service* on page 646
cluster-sp-session-registry.conf	*SP Session Registry Service* on page 643

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 stand-alone 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 663 for more information, available in the PingFederate distribution `sdk` directory.

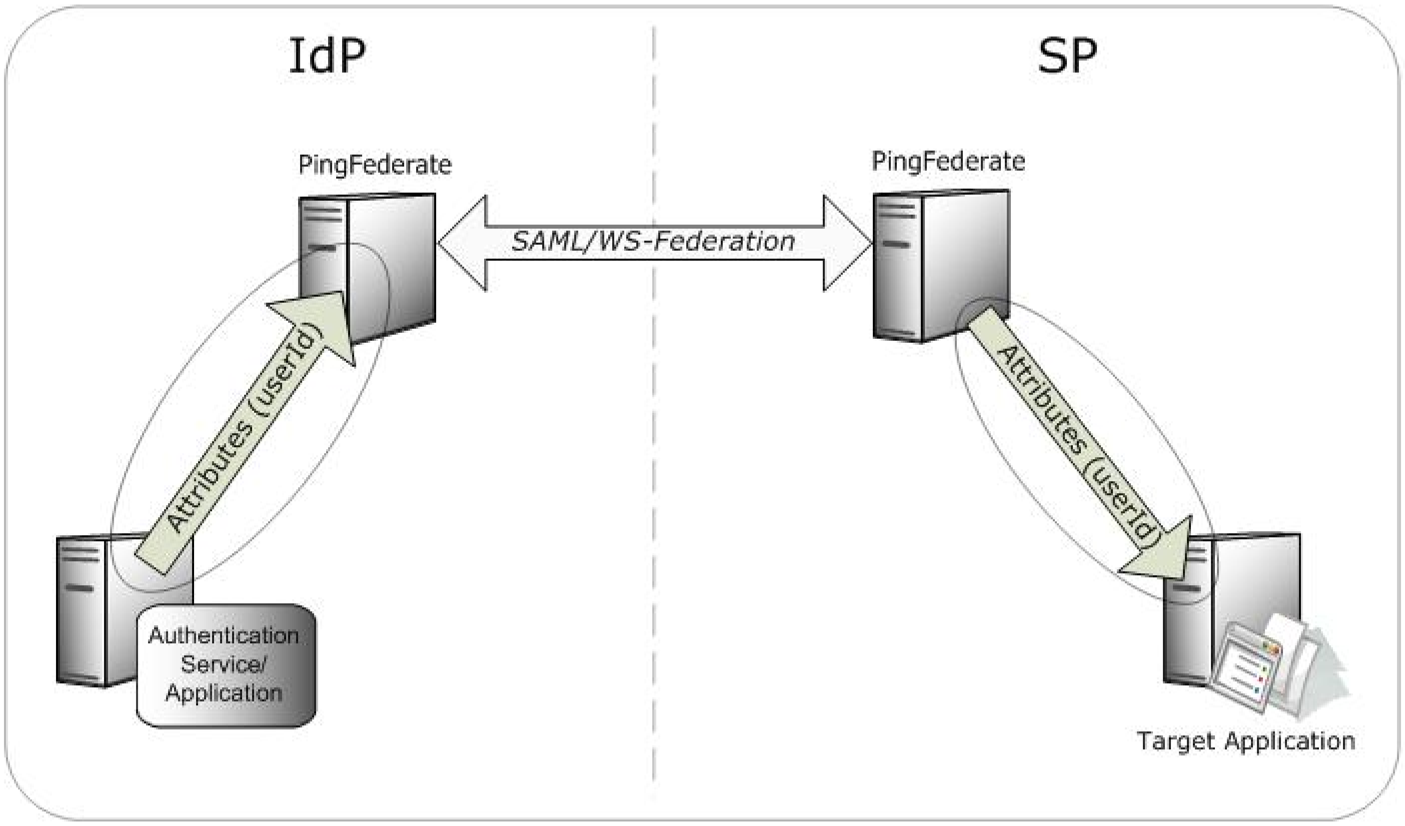
In addition, for integration with the PingFederate WS-Trust security token service (STS), we provide a range of *Token Translators*. These plug-in token processors (for an IdP) and token generators (for an SP) connect the STS with web service providers and clients for access to identity-enabled web services.

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 33.) 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.

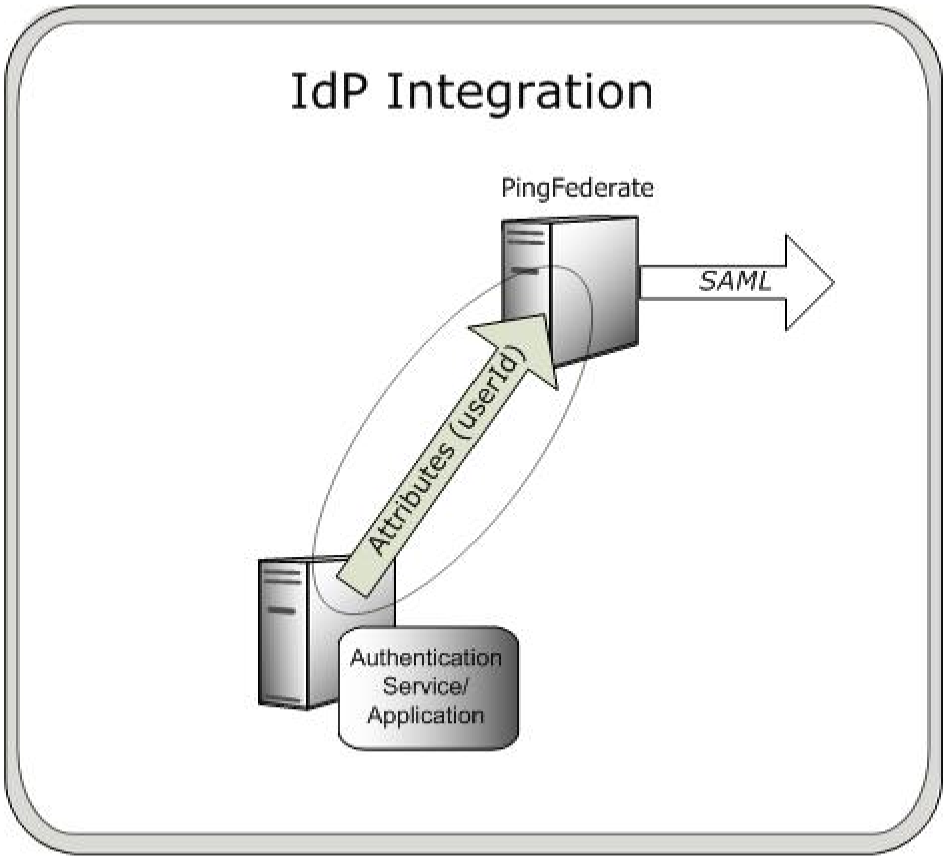
The following diagram illustrates the basic concepts of integration with PingFederate:



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.

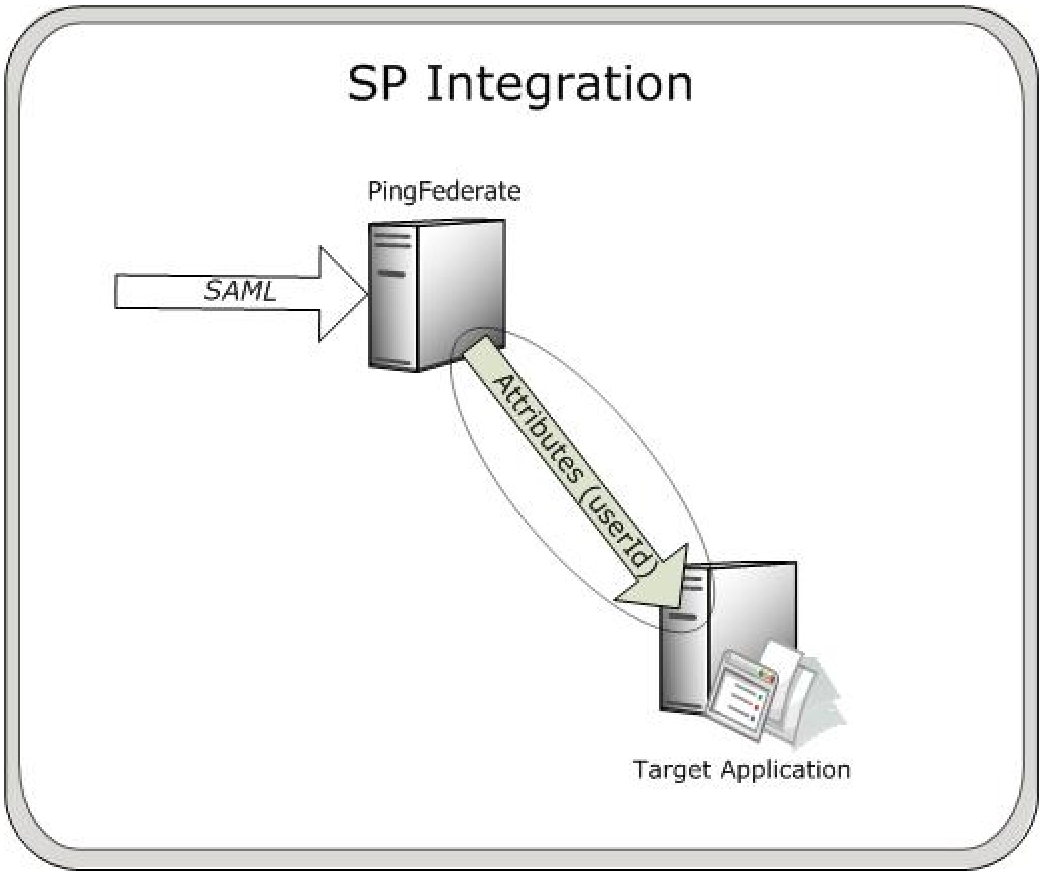
Authentication integration kits do not require any development; integration with PingFederate is accomplished entirely through the PingFederate administrative console. Ping Identity offers integration kits for authentication systems including:

- 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	.NET Integration Kit

Deployment scenarios	**IdP**	**SP**
	Agentless Integration Kit Java Integration Kit PHP 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 656*, described under *Additional Documentation* on page 663.)
- 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 656* 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 61* 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 following sections describe the directories and build components that comprise the SDK and provide instructions for setting up a development environment.

- *Directory structure* on page 665
- *Set up your project* on page 665

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

To start developing your own plug-in:

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 679.

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 following sections.

- *Configurable plug-in* on page 666
- *Describable plug-in* on page 666

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 `IdpAuthenticationAdapter` or 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 666, 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.

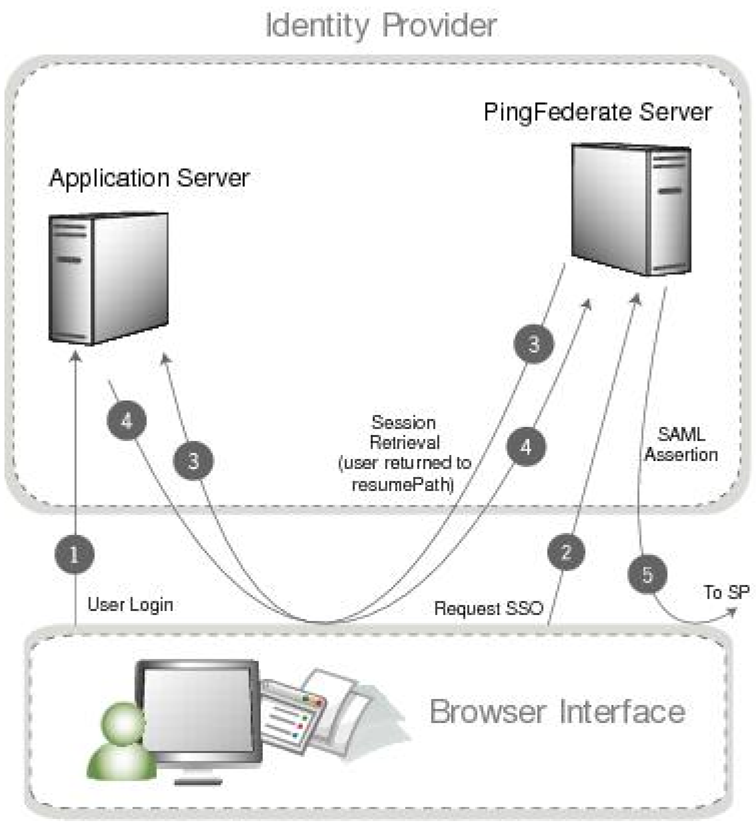
Tip: When creating a custom adapter, you can design it to render a template for processing and returning HTML to the user's browser using the `TemplateRendererUtil`. A sample `(template-render-adapter-example)` is included in the `sdk/plugin-src` directory of your PingFederate instance.

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:



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 666, 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.

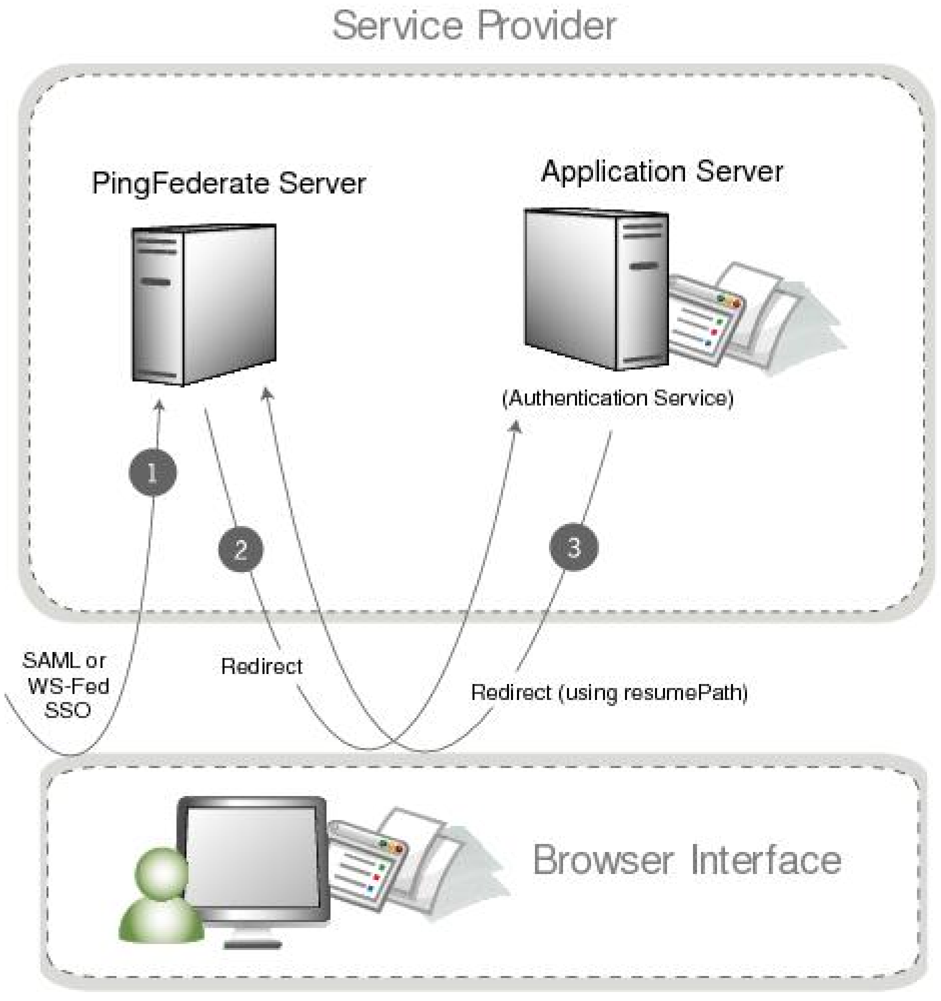
SP account linking

```
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 666, 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 666, 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 666, 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 666, 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 666, 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 676).

For each Identity Store Provisioner implementation, in addition to the methods described under *Shared interfaces* on page 666, 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, e.g. 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, e.g. 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, e.g. 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, e.g. 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, e.g. 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, e.g. the 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, e.g. 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, e.g. 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, e.g. 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, e.g. 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 674.

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 666, 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, e.g. 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, e.g. 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, e.g. 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, e.g. 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

The PingFederate Java SDK comes with an Apache Ant build script that makes building and deploying your project simple.

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 665).

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

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

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

Plug-in Type	Filename
Token Processor	token-processors
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

- 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

- 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.

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 9.1.4 - November 2018

PingFederate 9.1.4 is a cumulative maintenance release for PingFederate 9.1, which introduced many new features, such as authentication policy improvements, regional support for adaptive clustering, and OpenID Connect enhancements. For a full summary of the 9.1 release, see *PingFederate 9.1 - June 2018* on page 693.

Resolved issues

Ticket ID	Description
PF-21146	Resolved an issue where entering a long **Base URL** value on the **Server Configuration** > **Server Settings** > **Federation Info** screen could render the administrative console unresponsive.
PF-21144	Improved the cluster-protocol layer to only communicate with nodes running the same version of PingFederate.
PF-21064	The PingFederate SDK class `ConnectionSelectionFieldDescriptor` now supports searching of an IdP connection by its PingFederate-assigned id value.
PF-21021	Resolved an issue where PingFederate always returned the scope parameter in access token responses when the bypass authorization approval option is enabled for the client submitting the request or all clients. PingFederate now returns the scope parameter only when a resource owner chooses not to approve at least one requested scope.
PF-20998	The `connectionMetadata/convert` administrative API now only enables bindings for a connection when such information is included in the partner metadata.
PF-20967	Fixed an issue where PingFederate recorded an irrelevant warning message in the server log when handling token-issuance requests. The message read: `WARN [org.sourceid.oauth20.domain.PersistentGrantLifetimeHelper] Persistent Grant Lifetime is invalid (in minutes): null`
PF-20961	Resolved an LDAP attribute character-encoding issue for secondary data source lookups.
PF-20915	The adapterId parameter (case-sensitive) can now be used for self-service password reset.
PF-20914	The OAuth authorization endpoint no longer returns deleted exclusive scopes.

Ticket ID	Description
PF-20767	When an OpenID Connect (OIDC) IdP connection is an authentication requirement in an authentication policy, PingFederate now ensures that failures resulting from the OIDC IdP connection are handled in its **Fail** policy path.
PF-20765	Fixed an issue where PingFederate could not validate symmetrically-signed ID tokens that it received through OIDC IdP connections.
PF-20723	For OIDC protocol compliance, authorization requests must include the redirect_uri parameter; this is now the default behavior in new PingFederate installations. For upgrades, 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.
PF-20633	An HTML Form Adapter instance now allows users to change their passwords so long as it is configured with an instance of a password credential validator that supports changing passwords or an external password management system.

Upgrade considerations

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

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 *Manage cipher suites* on page 149 for more information.

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 153.

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 762.

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 older) or 9.0.2 to version 9.1.4, 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 9.1.4, 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 758.

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 758.

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 756 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 758.

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 647.

Metadata signing

Previously, when no signing certificate was chosen in the **Server Configuration** > **Server 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 **Server Configuration** > **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 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 4.0 (or older) drivers, 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 758.

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 761 for instructions.

Note: PingFederate has been tested with vendor-specific JDBC 4.1 drivers. 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 758 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 758.

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 758).

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 498.

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 758.

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 higher) 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 (SAML 2.0)* and *Choose an encryption certificate (SAML 2.0)*).

Note: The JCE provider libraries distributed with Gemalto SafeNet Luna SA HSM does not support the RSA-OAEP algorithm at the time of the PingFederate 7.1 R2 release. As a result, RSA-v1.5 is still allowed when deploying PingFederate with a Gemalto SafeNet Luna SA HSM.

OAuth persistent grants expiration

When upgrading to PingFederate 6.8 or higher, 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 higher) 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.

Deprecated features

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 101.

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 756.

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 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 4.0 (or older) drivers, 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 763.

JMX monitoring support for outbound provisioning

As of PingFederate 7.2, the JMX monitoring support for outbound provisioning has been deprecated and replaced with an audit file approach that is more inline with other auditing services offered by PingFederate.

JMX monitoring for outbound provisioning can be re-enabled in PingFederate 7.2 but will be removed in a future release (see *Monitor outbound provisioning* on page 158 for more information).

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.

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

Administrative console and administrative API

- 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

When using PingFederate with the Thales nShield Connect HSM, it is not possible to use an elliptic curve (EC) certificate as an SSL server certificate.

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.
- Adapter instances specified as **Sufficient** in a Composite Adapter configuration should be limited to adapter types that explicitly return control to PingFederate after a failure. Otherwise, the next adapter instance in the authentication "chaining" sequence (if any) may not be tried, and other unexpected behavior may occur. The following adapters work correctly under the **Sufficient** authentication policy in failure mode:
  - HTML Form
  - HTTP Basic
  - Kerberos
  - CoreBlox
  - Duo Security
  - IWA (returns control to PingFederate only if the failure is the result of invalid credentials after the configured number of retries)
  - OpenID (Generic)
  - OpenID (Google)
  - RSA
  - X.509

  Note that this list is updated as other adapters are modified, tested, and released.

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 custom 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.

Previous releases

PingFederate 9.1.3 - September 2018

PingFederate 9.1.3 is a cumulative maintenance release for PingFederate 9.1, which introduced many new features, such as authentication policy improvements, regional support for adaptive clustering, and OpenID Connect enhancements. For a full summary of the 9.1 release, see *PingFederate 9.1 - June 2018* on page 693.

Resolved issues

Ticket ID	Description
PF-20553 and PF-20551	Enhanced security for the Auditor accounts in the administrative console.
PF-20673	Fixed an issue where the SCIM inbound provisioning process skipped the subsequent custom SCIM attributes (if any) after processing an attribute with a null value.

Ticket ID	Description
PF-20463	When generating a certificate signing request (CSR) for a self-signed certificate, PingFederate now includes subject alternative names (SANs) if such are configured in the signed certificate.
PF-20195	Improved the access-token validation process when handling requests sent to the OAuth UserInfo endpoint (`/idp/userinfo.openid`).
PF-19360	Resolved an issue where JIT provisioning escaped characters incorrectly, which could introduce issues looking up provisioned accounts.

PingFederate 9.1.2 - August 2018

PingFederate 9.1.2 is a cumulative maintenance release for PingFederate 9.1, which introduced many new features, such as authentication policy improvements, regional support for adaptive clustering, and OpenID Connect enhancements. For a full summary of the 9.1 release, see *PingFederate 9.1 - June 2018* on page 693.

Resolved issues

Ticket ID	Description
PF-20119	Resolved a security issue in the Kerberos Token Processor. For more information, visit the Ping Identity *Support* website and refer to the Security Advisory SECADV017.
PF-20004, PF-20002, PF-20001	Enhanced the audit log to support an upcoming release of the PingFederate App for Splunk, version 2.0.
PF-19978	Upgraded to Jetty 9.3.24.
PF-19912	Resolved an issue where some certificates could not be imported into the global trust list on the **Server Configuration** > **Trusted CAs** screen.
PF-19851	Fixed an issue where the Simple Username Password Credential Validator failed to authenticate users whose records had been created or modified in a child instance of the validator.
PF-19808	Session management at the OAuth grant management endpoints for resource owners has been enhanced to take advantage of the timeout values defined under **Application Sessions** on the **Identity Provider (or SP Provider)** > **Sessions** screen.
PF-19799	Resolved an issue where the OAuth Access Grant Management Service retuned an HTTP status code 500 for invalid credentials. (It now returns 401 Invalid Credentials.)
PF-19060	PingFederate now records the cause of errors as opposed to always recording all errors as *Invalid Virtual Server ID* in the server log when the following conditions are met: • An SSO request is sent to a connection that has been configured with one or more virtual server IDs. • The request triggers an authentication policy and reaches a closed-ended policy path that ends with an authentication policy contract. • The in-policy mapping configuration of the authentication policy contract has been configured with one or more issuance criteria. • The request fails one of the issuance criteria.

PingFederate 9.1.1 - June 2018

PingFederate 9.1.1 is a cumulative maintenance release for PingFederate 9.1, which introduced many new features, such as authentication policy improvements, regional support for adaptive clustering, and OpenID Connect enhancements. For a full summary of the 9.1 release, see *PingFederate 9.1 - June 2018* on page 693.

Resolved issues

Ticket ID	Description
PF-19852	Resolved an issue where the administrative console limited the ability to map into authentication policy contracts and local identity profiles, and to chain the user identifier from an earlier authentication source in the policy.
PF-19831	Resolved an issue where PingFederate could fail to send SOAP back-channel requests (for example, the resolution requests for connections using the SAML 2.0 Artifact binding) through a forward proxy server.
PF-19608	Resolved an issue where URI fragment passed into the TargetResource parameter during SLO was not handled correctly.

PingFederate 9.1 - June 2018

Enhancements

Authentication

Authentication policy administration

Administrative UI improvements have been made to improve management of complex authentication policies involving many steps. This includes improved navigation of authentication policies and the ability to name and describe them.

Security levels for authentication sources

When supporting multiple authentication methods, some authentication methods will be equally secure or more secure than others. In these scenarios, it may not be necessary to challenge users for new credentials when they have an existing authentication session based on an equal or stronger level of authentication than is required for a new SSO or OAuth transaction. PingFederate now allows customers to group authentication sources, IdP adapter instances and IdP connections, by security level using a new authentication selector. This selector will branch authentication policy based on the authentication sources for which a user already has a session.

OAuth Client Set Authentication Selector

Applications may have differing needs for how users should be identified and authenticated. To define authentication policy behavior on a per OAuth-enabled application basis, PingFederate now includes the OAuth Client Set Authentication Selector.

Email ownership verification

Knowing a user is the owner of an email address is critical for use cases like forgotten username recovery where the user is emailed their username. Email ownership verification can be enabled when using local identity profiles. PingFederate tracks email ownership in a directory attribute, which can be used by PingFederate, such as for the purpose of allowing users to recover their usernames, and be released to downstream applications.

The URLs for email verification, and one-time links for self-service password reset for that matter, are resilient to restarts and secured using rotatable system keys.

Forgot username

Users who forget their username now have an option to recover it. This capability is added to the HTML Form Adapter to allow for self-service. Users enter their email address and, depending on the email ownership verification policy, receive an email with their username.

Connected identity attribute storage

All attributes received as a result of a user connecting a third-party identity to a local identity can be stored in the directory. These attributes can also be updated during a sign on event with that provider. These policies are enabled in local identity profiles.

CAPTCHA for HTML Form Adapter

CAPTCHA has been added to improve the prevention of automated systems invoking the HTML Form Adapter for user authentication, change password and account recovery.

Direct URL for change password in HTML Form Adapter

Your use case may require that users are able to change their password at PingFederate's HTML Form Adapter without forcing them through an SSO flow. This is now possible using a direct URL.

Direct URL for forgot password in HTML Form Adapter

Likewise, it is now possible to send users directly to the forgot password feature of the HTML Form Adapter with a direct URL.

HTML Form username pre-population based on WS-Federation authentication request

Usernames that are provided in the WS-Federation request, such as from Microsoft Azure AD, can now be pre-populated in the HTML Form Adapter's login UI so the user does not need to enter it.

Read-only fields

User's can be presented with a read-only view of certain attributes on the self-service profile management page.

Set password when no password exists

Users can now set a password on the self-service profile management page when they originally registered without one, such as registration via a third-party identity provider.

Single sign-on

Signing SAML response and assertion per connection

SAML service providers have the option to require both the SAML assertion as well as the SAML response to be signed. PingFederate now allows for this redundant signing to be configured per connection.

Multiple issuers from a single OpenID Connect Provider

PingFederate now supports multiple OIDC issuers in a single IdP connection to better support multi-tenant OIDC Providers, such as Microsoft Azure AD.

OAuth

ID Token customization

Some OpenID Connect (OIDC) use cases may require customizing standard ID Token claims, such as overriding the iss claim to something other than the PingFederate Base URL. This is now possible by mapping standard ID Token claims through contract fulfillment in the OpenID Connect Policy.

ID Token vs. UserInfo attributes

OIDC defines options for delivering user attributes to Relying Parties through a user's browser via an id_token or based on a back-channel call to the UserInfo endpoint. Many user attributes are sensitive so you may want to control which attributes are delivered through the user's browser and which are delivered only through the UserInfo endpoint. PingFederate now gives you this control at a per-attribute basis. This is additional functionality beyond the existing capability where all attributes can be included in the ID Token.

Persistent grant lifetime based on context of authorization request

The persistent grant lifetime is the amount of time for which a user's consent to an OAuth client is valid. PingFederate has support for a default lifetime and the ability to override the lifetime based on the OAuth client. However, it may be necessary to control this lifetime based on the context of the authorization event, such as by requested scopes or user attributes resulting from authentication. PingFederate now has the flexibility to set the lifetime of each approved persistent grant based on context of the authorization request.

OpenID Connect metadata customization

OIDC metadata is used to share protocol level details publicly. You may want to limit or override the data that PingFederate would normally publish at the /.well-known/openid-configuration endpoint. This is now possible using a template to dynamically control all the details returned to requesting clients.

Disable OAuth clients

It's now possible to disable an OAuth client. PingFederate will respond to authorization requests or grant requests for a disabled client in the same way it would if the client was not defined in the system. This can be useful in the event that an OAuth client's access needs to be temporarily blocked.

System

Geographically distributed clustering

Customers with global user populations, and who deploy PingFederate clusters across data centers, have had to learn complex concepts for the configuration required to properly manage runtime state, such as authentication sessions. We've added new capabilities to make this much simpler, and they build on top of Adaptive Clustering (available as of PingFederate 9.0). Depending on the type of state, PingFederate will either proactively replicate or retrieve the state across regions on-demand. Generally, longer lived state, such as an authentication session, will be proactively replicated to minimize disruptions to end users in the event of a data center outage. Shorter lived state, such as an OAuth authorization code, will be retrieved on-demand across regions to support wandering users or where the user and the application are in different regions. All of this is managed automatically by simply specifying a region identifier for each runtime engine.

Support for systemd services

The Linux installer now supports installation of systemd services on Red Hat Enterprise Linux (RHEL) 7, while maintaining System V service support on RHEL 6. The systemd init system offers more robust service management and a simpler configuration. When upgrading PingFederate on a system that supports systemd, you will be given the option to replace an existing System V service with a newer systemd service. A template service unit configuration file is also provided for manual installation of systemd services on non-RHEL systems.

LDAP Service (SRV) records

Customers with complex LDAP environments, where many servers are distributed across data centers, can enable LDAP server lookup using SRV records. SRV records are used to map the name of a service to a DNS name.

X.509 certificate subject alternative name

To continue to support browsers that are shifting requirements for identifying the name of a server by using subject alternative name (SAN) rather than common name (CN), customers can now create certificates and certificate signing requests (CSRs) with SANs.

Cluster node shutdown sequencing

Nodes in an adaptive cluster that are shutdown will stage the shutdown process to properly retain runtime state.

HTTP header logging

HTTP headers can now be logged in the audit and server logs to improve transaction debugging and reporting.

Administrative API enhancements

The administrative API has been extended to include the following workflows:

Adapter and redirect configuration

- `/redirectValidation` to manage trusted URLs that PingFederate is allowed to redirect the browser to.

- `/sp/authenticationPolicyContractMappings` to manage mappings between authentication policy contracts and SP adapter instance.
- `/idp/spConnections` enhanced to support connection-level adapter overrides in SP connections.
- `/sp/idpConnections` enhanced to support connection-level adapter overrides in IdP connections.

OAuth and OpenID Connect

- `/keyPairs/oauthOpenIdConnect` to manage static keys for OAuth and OpenID Connect.
- `/oauth/clientSettings` enhanced to support the OAuth 2.0 Dynamic Client Registration protocol.

Outbound provisioning

- `/serverSettings/outboundProvisioning` to manage general outbound provisioning settings.
- `/idp/spConnections` enhanced to support outbound provisioning in SP connections.

WS-Trust STS

- `/idp/stsRequestParametersContracts` to manage WS-Trust STS request contracts.
- `/idp/tokenProcessors` to manage token processor instances.
- `/sp/tokenGenerators` to manage token generator instances.
- `/tokenProcessorToTokenGeneratorMappings` to manage mappings between token processor instances and token generator instances.
- `/idp/spConnections` enhanced to support WS-Trust STS settings in SP connections.
- `/sp/idpConnections` enhanced to support WS-Trust STS settings in IdP connections.

Other improvements

Security enhancement

The SAML 2.0 Attribute Query Service no longer accepts requests that send the SharedSecret parameter in a query string. We recommend to only pass in this sensitive parameter in the message body via the HTTP POST method.

Updated components

The following bundled components and third-party dependencies have been updated:

- PingID Integration Kit 2.2
- UnboundID LDAP SDK 4.0.5
- Apache Commons FileUpload 1.3.3
- Apache Commons Validator 1.6
- Jackson 2.7.9.3
- Jetty 9.3.23
- JGroups 3.6.15
- jose4j 0.6.3
- Log4j 2 2.11
- Spring Framework 4.3.16

Resolved issues

Ticket ID	Description
PF-19823 and PF-19748	Resolved a couple of issues where integration with a Gemalto SafeNet Luna SA HSM failed in a clustered PingFederate environment.
PF-19806	Fixed an issue where an access token could become invalid shortly after it was issued when an OAuth authorization request involved an access token management (ATM) instance and an IdP connection with the following characteristics:

Ticket ID	Description
	• The ATM instance was configured to verify the validity of the resource owner's authentication sessions as part of the access-token validation process (see *Manage session validation settings* on page 305). • PingFederate was configured to track authentication sessions for the IdP connection (see *Sessions* on page 261).
PF-19792	Resolved a security issue in the area of customer identity and access management. For more information, visit the Ping Identity *Support* website and refer to the Security Bulletin SECBL010.
PF-19655	Resolved a validation issue that prevented setting two-digit node indexes in the IWA IdP Adapter instance configuration.
PF-19490	The administrative API no longer requires a configuration value for any non-mandatory, radio button-style setting.
PF-19236	Resolved an issue where PingFederate did not record the client ID in the audit log when the client failed to provide proper credentials for authentication.
PF-19200	Fixed an issue where PingFederate could not to join an existing cluster if its pf.cluster.tcp.discovery.initial.hosts property was only configured with a partial list of peer nodes, despite the fact that these nodes were operational and reachable.
PF-19117	Resolved an issue where the `/pf/heartbeat.ping` endpoint could return an "OK" browser message and an HTTP status code 200 when the server was not ready to handle other runtime requests.
PF-19071	Administrators may now specify the maximum amount of time to wait before a JDBC query times out. The configuration file is `log4j2.db.properties`, located in the `<pf_install>/pingfederate/server/default/conf`. The property is defaultQueryTimeout. When no value is configured (the default behavior), the timeout value from the JDBC driver associated with the target database server is used. **Note:** defaultQueryTimeout is a global setting that applies to all JDBC queries, to all qualified database servers.
PF-18981	PingFederate now records in the server log the errors encountered when attempting to fulfill attribute values via data stores.
PF-18675	Improved the efficiency in persistent grant queries.
PF-18194	Fixed an issue where the HTML Form Adapter ignored the **Session Timeout** value and prompted users to authenticate every SSO request when the following conditions were met: • The adapter instance was configured to authenticate against a RADIUS server via an instance of the RADIUS Username Password Credential Validator. • The adapter instance was configured with a **Session Max Timeout** field value.
PF-17928	Resolved an issue where the **Change Password** link was not always visible when the **Password Management System** was configured in the invoking HTML Form Adapter instance.
PF-17917	When handling an OAuth request that comes with a max_age parameter value, if the authentication source does not provide the time of authentication (also known as the "authentication instant"), PingFederate prompts the user to reauthenticate once (the expected behavior) and records the event in the server log (the improvement made in this release).

Ticket ID	Description
	The HTML Form Adapter returns the time of authentication by default (and configurable by the **Track Authentication Time** advanced setting). For custom IdP adapters developed using the PingFederate SDK, refer to the Javadoc for the `IdpAuthenticationAdapter` interface for more information. **Tip:** The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.
PF-17842	Cipher suites are now selected based on the order they are listed in the configuration file (see *Secure sockets layer* on page 76).
PF-17815	When a PingDirectory™ user initiates a password reset request, the HTML Form Adapter no longer prompts the user to reset password twice.
PF-17518	Resolved an issue where PingFederate did not record in the audit log the reason of an issuance-criterion failure when the following conditions were met: • An issue criterion was configured in an authentication source. • The authentication source was configured as a checkpoint in an authentication policy. • A request triggered the authentication policy and failed the issue criterion configured in the authentication source.
PF-17507	Updated the input validation logic to broaden the characters that can be used to define valid paths on the **Server Configuration** > **Redirect Validation** screen.
PF-17140	Custom messages returned form Password Credential Validator implementations supporting self-service password reset are now displayed to users when an error occurs.
PF-17059	For IdP adapters that do not return authentication time information, the creation time of an associated authentication session will be relied upon to fulfill protocol details such as the OpenID Connect auth_time claim.
PF-16884	Fixed an issue where the session revocation endpoint (`/pf-ws/rest/sessionMgmt/revokedSris`) did not work when the pf.runtime.context.path property was set in the `run.properties` file.
PF-16323	Resolved a Salesforce outbound provisioning issue where the **Attribute Mapping** configuration screen could become unresponsive and the provisioning process could result in error when PingFederate receives from Salesforce duplicate values for fields defined on the **Attribute Mapping** screen. Both the administrative console and the provisioner have been enhanced to handle this scenario. Additionally, such conditions are now recorded as `DEBUG` messages in the server log and the provisioner log.
PF-16317	The `/dataStores` administrative API endpoint no longer returns the type attribute twice in its response.
PF-16284	Improved the way that cluster messages are handled for better response time.
PF-15202	If a setting or a block of settings has been removed from a configuration file stored in the `<pf_install>/pingfederate/server/default/config-store` directory, the upgrade tools preserves such customizations by removing such setting or block of settings from the new installation and records the removals in its log file (see *Copy customized files or settings* on page 756).

PingFederate 9.0.4 - May 2018

PingFederate 9.0.4 is a cumulative maintenance release for PingFederate 9.0, which introduced many new features, such as adaptive clustering, OAuth dynamic client registration, LDAP directory for OAuth client storage, cross-origin resource sharing (CORS) for OAuth endpoints, consumer authentication, registration and profile management. For a full summary of the 9.0 release, see *PingFederate 9.0 - December 2017* on page 703.

Resolved issues

Ticket ID	Description
PF-19624	Resolved an issue where requests leveraging an OAuth IdP connection grant-mapping configuration could fail when authentication sessions were enabled.
PF-19546	Fixed an issue where the colon character could not be used in scope values when PingFederate was configured to store OAuth client records in a directory server.
PF-19544	When PingFederate receives an administrative API request at `/oauth/openIdConnect` for the purpose of configuring an optional association between a standard scope and attributes in an OpenID Connect policy, it no longer returns an error message if a related standard attribute does not exist for the standard scope that the association is being configured.
PF-19523	Resolved an issue where PingFederate could fail to retrieve the node index registry on startup when joining a cluster under load, which could lead to the inability to retrieve session-state information that relied on the resolution of node indexes to cluster members at a later time.
PF-19443	When handing the scenario where an SP sends a SAML 2.0 AuthnRequest message with an ACS endpoint and the user fails to fulfill the authentication requirements, PingFederate (as the IdP) now returns the SAMLResponse message to the provided ACS endpoint if one of the following conditions is met: • The provided ACS endpoint matches one of the ACS endpoints defined in the corresponding SAML 2.0 SP connection. • The provided ACS endpoint does not match any the ACS endpoints defined in the corresponding SAML 2.0 SP connection but the AuthnRequest message is digitally signed by the SP and PingFederate is able to verify the digital signature.
PF-19398	Fixed an issue where PingFederate sent redundant LDAP queries to the directory server for each outbound provisioning cycle.
PF-19306	Resolved an issue where PingFederate failed to handle virtual attributes that had been added to local identity profiles.
PF-19230	When processing client registrations received at the dynamic client registration endpoint, PingFederate no longer ignores the jwks client metadata when the metadata value is a valid JSON object.
PF-19213	Resolved an issue where PingFederate might fail to process requests when it was configured to record log messages to a database server and the database server was unreachable.
PF-19106	When PingFederate receives an administrative API request via the HTTP GET method at `/authenticationPolicies/default`, it no longer includes in its response any inboundMapping configuration data as part of the LOCAL_IDENTITY_MAPPING configuration, for which neither registration nor profile management is enabled. Similarly, when processing a PUT request sent to `/authenticationPolicies/default`, PingFederate does not require any inboundMapping configuration data

Ticket ID	Description
	as part of the LOCAL_IDENTITY_MAPPING configuration, for which neither registration nor profile management is enabled.
PF-19078	Resolved a memory leak issue that could increase response time in a clustered PingFederate environment after configuration had been replicated hundreds of times.
PF-19061	When an OAuth request fails, PingFederate returns to the client an error message, which contains an error field and an error_description field. PingFederate no longer includes irrelevant information as part of the error_description field value.
PF-17823	Fixed an issue where PingFederate recorded false positive `ERROR` messages in the server log on startup.

PingFederate 9.0.3 - March 2018

PingFederate 9.0.3 is a cumulative maintenance release for PingFederate 9.0, which introduced many new features, such as adaptive clustering, OAuth dynamic client registration, LDAP directory for OAuth client storage, cross-origin resource sharing (CORS) for OAuth endpoints, consumer authentication, registration and profile management. For a full summary of the 9.0 release, see *PingFederate 9.0 - December 2017* on page 703.

Resolved issues

Ticket ID	Description
PF-19014	Resolved an issue where errors could occur as PingFederate removed expired artifacts.
PF-18907	When PingFederate receives an erroneous authorization request, such as an authorization request from an unknown client, it now records the error conditions in the security audit log. Values for the event, role, and status fields are `OAuth`, `AS`, and `failure`, respectively. Error messages are captured under the description field.
PF-18865	The `/pf-scim/v1/Schemas` endpoint now returns the HTTP status code 401 when client applications provide no (or invalid) authentication credentials.
PF-18853	Resolved an issue where the `oauth-client-management-sqlserver.sql` table-setup script did not create a proper index on the IDX_FIELD_NAME column since version 8.4. (For customers who had created such index using such table-setup script, it is recommended to remove the IDX_FIELD_NAME index and recreate it. For more information, please consult with your database administrators and refer to the new table-setup script, located in the `<pf_install>/pingfederate/server/default/conf/oauth-client-management/sql-scripts` directory.)
PF-18802	An OAuth client can now be saved with a JWKS URL (or a JWKS) regardless of whether the client is configured to use signed JWTs for client authentication or transmission of request parameters (or for both purposes). Additionally, when PingFederate AS receives an authorization request with a signed request object from a client that has not been configured with a JWKS URL or JWKS, PingFederate now ignores the signed request object and processes the request based on request parameters found outside of the signed request object.
PF-18801	The **OAuth & OpenID Connect Keys** configuration no longer enforces that an active key must be selected for each algorithm. For example, if an organization only wishes to support RSA algorithms, its administrators are no longer required to create and select an active key for any EC algorithms.

Ticket ID	Description
PF-18614	Fixed an interoperability issue when static OAuth and OpenID Connect keys were enabled.
PF-18464	Resolved an issue where PingFederate might not finish its startup sequence when multiple PingFederate installations were deployed using adaptive clustering.
PF-18443	The PingFederate IdP server now records the SP's entity ID and the protocol information for authentication attempts sent to any WS-Federation SP connections.
PF-18399	Fixed an issue where users were not able to sign on to the profile management page if they had already authenticated and their sessions were tracked by PingFederate. (Note that *authentication-session tracking* is optional and disabled by default.)
PF-18327	Restored the functionality of allowing custom authentication selectors that were designed to return an IdP adapter ID or the connection ID of an IdP connection to be used in authentication policies. Some additional steps are required. For more information, see *Upgrade considerations* on page 683.
PF-18275	Resolved an issue where an attribute in an access token attribute contract could be malformed when it was mapped from the scope of the request.
PF-17971	Resolved an issue with self-service account recovery for usernames containing special characters. The root cause was unnecessary HTML encoding on the username before a password credential validator processes it. This encoding has now been removed.
PF-16977	If PingFederate is set up as the identity bridge for PingOne® through a managed SP connection, and when changes that affects such connection have been made (for example, a new base URL or signing certificate), PingFederate now prompts for confirmation before uploading the changes to PingOne.

PingFederate 9.0.2 - February 2018

PingFederate 9.0.2 is a cumulative maintenance release for PingFederate 9.0, which introduced many new features, such as adaptive clustering, OAuth dynamic client registration, LDAP directory for OAuth client storage, cross-origin resource sharing (CORS) for OAuth endpoints, consumer authentication, registration and profile management. For a full summary of the 9.0 release, see *PingFederate 9.0 - December 2017* on page 703.

Resolved issues

Ticket ID	Description
PF-18452	Resolved an issue where upgrading to PingFederate Server 9.0 or 9.0.1 when using its provisioning capability could result in user records being disabled at SaaS applications. If you are upgrading from version 8.4.4 (or an earlier version) to PingFederate 9.0.2, 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 9.0.2, refer to *Upgrade considerations* on page 683 for more information.
PF-18384	Fixed a rare issue where engine nodes could not retrieve session-state information from the replica set when adaptive clustering was enabled.
PF-18380	Resolved an issue where PingFederate could not fulfill the persistent grant contract when the source was **AccountLink**.
PF-18318	Fixed a provisioning compatibility issue where the SaaS Connector that use a manager attribute lookup could not complete their (outbound) provisioning tasks.

Ticket ID	Description
PF-18190	The process of auto-generating cluster node index has been improved to accommodate network configurations where cluster bind addresses across regions could collide. The value now ranges from 0 to 2,147,483,647.
PF-18173	The outbound provisioner's pagination process has been improved to handle the situation where a directory server returns a large result set.
PF-16965	Upgraded to JGroups 3.6.9 to improve the performance of cluster communication.

PingFederate 9.0.1 - January 2018

PingFederate 9.0.1 is a cumulative maintenance release for PingFederate 9.0, which introduced many new features, such as adaptive clustering, OAuth dynamic client registration, LDAP directory for OAuth client storage, cross-origin resource sharing (CORS) for OAuth endpoints, consumer authentication, registration and profile management. For a full summary of the 9.0 release, see *PingFederate 9.0 - December 2017* on page 703.

Resolved issues

Ticket ID	Description
PF-18064	Upgraded to UnboundID LDAP SDK for Java 4.0.4 to resolve an issue where PingFederate could not establish LDAP connections when the target LDAP server was not configured to allow anonymous access to the root directory server agent service entry (root DSE).
PF-17936	Fixed an administrative user interface issue where default values could not be set for fields using the **Checkbox**, **Date**, or **Text** data type on the **Fields** screen when configuring a local identity profile.
PF-17911	Fixed an issue where connections and OAuth clients that were created and removed while one or more engine nodes were offline lingered on and prevented connections and clients using the same IDs to be created later.
PF-17863	Resolved an issue where the access-token session-validation process did not slide the idle timeout value upon successful validation.
PF-17790	As a federation hub, PingFederate no longer fails the second and subsequent SSO attempts sent to the `/sp/startSSO.ping` endpoint when the IdP connection is only associated with an authentication policy contract without any SP adapter instances and the **Enable Sessions for All Authentication Sources** check box is selected on the **Sessions** screen.
PF-17402	Fixed an issue where attribute values from an authentication policy contract were not mapped into access tokens when the authentication policy contract was fulfilled by an IdP connection.
PF-17112	The process of auto-generating cluster node index has been improved to accommodate network configurations where cluster bind addresses across regions could collide. The value now ranges from 0 to 16,777,215.
PF-16919	PingFederate now supports RSASSA-PSS digital signature algorithms (PS256, PS384, and PS512) in the following areas: • Signing JWT-based OAuth access tokens and the introspection of them. • Signing OpenID Connect ID tokens and verifying RSAPSS-signed ID tokens. • Signing private key JWTs and verifying RSAPSS-signed private key JWTs; for inbound and outbound OAuth client authentication and sending and receiving request objects.

Ticket ID	Description
PF-16071	PingFederate is now qualified on Thales nShield Connect 12.40 (firmware) using its client driver 12.40.2.

PingFederate 9.0 - December 2017

Enhancements

Adaptive clustering

Prior to PingFederate 9.0, our clustering mechanism allowed tweaking many different settings to optimize the environment based on your use cases and deployment requirements. While the mechanism is sophisticated, its directed nature can be cumbersome as you scale horizontally in an IaaS cloud, such as Amazon EC2. Adaptive clustering in PingFederate 9.0 solves this.

Adaptive clustering is very easy to use. It's one simple parameter. When enabled, clusters can elastically scale to any number of nodes. Each node is set with the same cluster configuration, so a single image is all that's needed. As demand increases, more instances of that image can be provisioned without cluster configuration customizations, regardless of the size of the existing cluster.

Customer identity and access management (CIAM)

Consumer grade authentication demands a smooth user experience. If your application is not easy, users will walk. In PingFederate 9.0 we've purpose-built authentication, self-service registration, and profile management to meet the expectations of consumers and made it simple to enable.

Users can now be given the option to sign on with a password, sign on via a third party identity provider, or self-service register. This is all integrated into a single, and completely brandable, UI.

The configuration is done by combining authentication policy contracts, the HTML Form Adapter, authentication policies, and a new configuration object called a local identity profile. The local identity profile is the glue that ties together what third party identity providers will be offered as alternate authentication and registration options, along with the fields that will be presented on the registration and profile management screens and how those fields map into the identity store.

The field configuration is very flexible. You can easily add any number of fields to the registration and profile screens, each with its own label and control type. We support text, check box, drop-down list, email, multi-select check box group, phone, and date control types. In the local identity profile you tie these fields to the identity store by mapping each field to a directory attribute.

Enabling third party identity providers will give users options to leverage an existing identity to register and sign on to your application. These are enabled by listing them on the local identity profile and mapping them through authentication policy. Any IdP connection or IdP adapter can be used as a third party. You can then map the attributes returned by that third party directly into the fields for the local identity profile to streamline user registration.

Lastly, multiple instances of local identity profiles can be defined, so you can offer users different combinations of third party identity providers, registration requirements, and branding, depending on the type of user or the applications that they may attempt to access.

OAuth Dynamic Client Registration

As OAuth deployments continue to mature, we are seeing continued interest in related standards. In this release we have added support for Dynamic Client Registration (*RFC7591*). This defines a standardized API that app developers can use to register their apps at OAuth authorization servers. We support this API as well as configurable policies for default configuration, a limit on which scopes can be included in the client registration, fine grain control over which clients can register, and client configuration that can be dynamically set based on context of the registration. This makes it easy for app developers to register their apps while giving the administrator the ability to maintain complete control.

Extended OAuth client metadata

Many customers need to track additional data related to OAuth clients beyond that which is required for runtime processing, which is what PingFederate has traditionally supported. Some customers need to track information like developer contact details, application type, department ownership, etc. We have added extensible metadata to OAuth clients to fill this need. You now have the ability to define an extended schema of metadata and to store values for that metadata with OAuth clients.

LDAP directory for OAuth client storage

To add to our existing storage options for OAuth clients, which includes the default option of on the file system, external database (JDBC) and custom data storage using our SDK, we’ve now added support for storing clients in a directory server using LDAP.

Cross-origin resource sharing (CORS) for OAuth endpoints

Configuring CORS is now drastically simpler. Administrators can enable or disable CORS by defining allowed origins configuration all within the administrative console, without needing to restart the server.

Multiple brand support for end user email templates

Customers supporting multiple user populations or multiple brands need to present a consistent brand experience to end users across all interactions. We have added support for customizing emails sent to end users per HTML Form Adapter instance. This builds on branding controls that already exist to customize the HTML templates served to end users when authenticating and maintaining passwords.

Additional OAuth and OpenID Connect enhancements

- **OAuth Client Credentials grant type attribute mapping**: Attributes can now be mapped into access tokens that are issued using the Client Credentials grant type.
- **OpenID Connect attribute mapping per scope**: It is now possible to explicitly map attributes to scopes for OpenID Connect. Prior to PingFederate 9.0 all custom attributes were returned using the profile scope. Administrators can now assign individual attributes to specific scopes.
- **OAuth and OpenID Connect static keys**: PingFederate rotates a set of keys used for OAuth and OpenID Connect signature operations, such as signing ID tokens and authentication requests. Customers now have the option to statically assign these keys and publish their associated certificate chain.
- **Publish separate base URLs for token and authorization endpoints**: The OAuth token endpoint is called directly by OAuth clients, such as a native mobile application, to request access tokens. The OAuth authorization endpoint is accessed by end users via browser to authorize clients. It is now possible to configure different base URLs for these two endpoints. This improves support for OAuth clients authenticating to the token endpoint using mutual transport level security authentication.
- **OAuth Access Grant API attributes**: The OAuth Access Grant Management Service has been enhanced to return all attributes stored with the grant.
- **OAuth response type constraints**: Allowed response types can now be restricted per OAuth client. This controls what tokens can be returned from the authorization endpoint. A dynamic client registration policy is also included to enforce response type constraints for clients that register via the dynamic client registration API.
- **State hash for ID token**: Support has been added for setting the state hash (s_hash) value in the ID token to allow OAuth clients to verify that the state is the same as what was included in the authentication request.

Security enhancement

A security enhancement has been made to safeguard JDBC data store queries against back-end SQL injection attacks. (For more information, see *Upgrade considerations* on page 683).

Other improvements

- Enhanced LDAP failover capability. When multiple host names or IP addresses are provided for a given LDAP data store, in addition to network error conditions, PingFederate now also fails over to the next server if the current server returns an LDAP system error.

- The following bundled components and third-party dependencies have been updated:
  - PingID Integration Kit 1.5
  - Jetty is 9.3.21
  - jose4j is 0.6.1
  - Spring Framework 4.3.10

Resolved issues

Ticket ID	Description
PF-17771	Resolved an issue where the `provmgr` command-line utility failed when the source user repository was connected via LDAPS.
PF-16923	Fixed an outbound provisioning configuration issue when the only protocol selected under the **Identity Provider** role was **Outbound Provisioning**.
PF-16860	If a WS-Trust 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.
PF-16851	In some use cases, administrators can optionally chain data store queries, using results from previous queries as inputs for a subsequent query. An issue was found and resolved where the administrative console did not display the configuration properly. Note that runtime transactions were never adversely affected by this user interface issue.
PF-16827	The `/authenticationPolicies/default` administrative API endpoint has been improved to handle gracefully the error condition where an update references one or more invalid IdP adapter instance IDs.
PF-16760	Some adapters require administrators to upload multiple configuration files. An issue was found and resolved where the administrative console did not display the file names of all the selected files.
PF-16749	PingFederate now uses the http.soLinger parameter consistently in the following configuration files: • `<pf_install>/pingfederate/bin/start.ini` • `<pf_install>/pingfederate/etc/jetty-admin.xml` • `<pf_install>/pingfederate/etc/jetty-runtime.xml`
PF-16545	Resolved an issue where the accessGrantUpdated field values were not updated when access grants were stored on an LDAP server.
PF-16529	Resolved an issue where introspection of access tokens could fail under some conditions.
PF-16512	Fixed an issue where the PingFederate SCIM endpoints returned an error response when a request asked for a list of id attribute values alone.
PF-16413	The user interface for Connection Set Authentication Selector now sorts the list of available connections for a better administrative experience.
PF-16262	Users are always redirected to the sign-on screen for authentication after changing their passwords via the HTML Form Adapter, regardless of whether PingFederate is deployed in a standalone environment or a clustered environment.

Ticket ID	Description
PF-16100	When a user tries to reset a password but enters an invalid username, PingFederate now records the attempt in the audit log as a failure with the error condition.
PF-16097	After initiating a configuration replication, administrators are no longer required to restart the PingFederate service on the engine nodes in order for the engine nodes to activate the changes that have been made to the `<pf_install>/pingfederate/server/default/data/config-store/http-request-parameter-validation.xml` file on the console node prior to the replication.
PF-16070	The policy engine has been fine-tuned for better performance when handling complex authentication policies.
PF-15814	If runtime notification is configured, PingFederate now sends an email notification once for each expiring certificate on a per-connection basis.
PF-13704	Simplified log messages for clustering events.

Versions 8.x and 7.x

PingFederate 8.4.4 - November 2017

PingFederate 8.4.4 is a cumulative maintenance release for PingFederate 8.4, which introduced many new features, such as configuration at scale, OpenID Connect signed requests, JWT OAuth 2.0 authorization grants and client authentication, dynamic discovery of cluster nodes using AWS Roles, self-service account unlock, Docker support, and more. For a summary of the 8.4 release, see *PingFederate 8.4 - June 2017* on page 709.

Resolved issues

Ticket ID	Description
PF-17472	Fixed an issue where the administrative API failed to update an IdP connection when it was configured with account linking and OAuth attribute mapping.
PF-17349 and PF-17330	Resolved two rare deadlock issues.
PF-17215	Resolved an issue where a configuration replication error could occur when a previously removed OAuth client was recreated with the same client ID.
PF-17127	Introduced a new setting (InterReqStateMgmtMapImpl.max.session.attrs in the `size-limit.conf` file) to optimize the memory usage of the Inter-Request State-Management Service.
PF-17108	Resolved an issue where the Kerberos Adapter could not fail over to the next adapter when the pf.runtime.context.path property was configured in the `run.properties` file.
PF-17035	Fixed an issue where the Composite Adapter could skip the OpenToken Adapter's **Logout Service** URL when processing logout requests.
PF-17009	Resolved an issue where expired grants were not cleaned up when they were stored on an LDAP server.

PingFederate 8.4.3 - October 2017

PingFederate 8.4.3 is a cumulative maintenance release for PingFederate 8.4, which introduced many new features, such as configuration at scale, OpenID Connect signed requests, JWT OAuth 2.0 authorization grants and client authentication, dynamic discovery of cluster nodes using AWS Roles, self-service account unlock, Docker support, and more. For a summary of the 8.4 release, see *PingFederate 8.4 - June 2017* on page 709.

Resolved issues

Ticket ID	Description
PF-17097	Resolved an issue where the outbound provisioning engine could stop and subsequently wrote a "Timer already cancelled" error message to the provisioner log.
PF-17053	Fixed an issue where SSO requests might fail when connections were configured with one or more ReferenceID Adapter instances with per-connection overridden instance settings.
PF-16997	If an SSO request invokes a target URL-to-SP adapter instance entry (configured on the **Service Provider** > **Target URL Mapping** screen), and if the resulting SP adapter instance is one of the multiple target session mappings with per-connection overridden instance settings, PingFederate now honors the overridden values at runtime.

PingFederate 8.4.2 - September 2017

PingFederate 8.4.2 is a cumulative maintenance release for PingFederate 8.4, which introduced many new features, such as configuration at scale, OpenID Connect signed requests, JWT OAuth 2.0 authorization grants and client authentication, dynamic discovery of cluster nodes using AWS Roles, self-service account unlock, Docker support, and more. For a summary of the 8.4 release, see *PingFederate 8.4 - June 2017* on page 709.

Resolved issues

Ticket ID	Description
PF-16908	Fixed an issue where the Kerberos Adapter returned the Username attribute value as that of the Domain/Realm Name attribute.
PF-16878	Resolved an issue where some OAuth clients that had been granted access to an Access Token Management instance in the past might lose such privilege after an upgrade.
PF-16876	Upgraded to Jetty 9.3.20 to resolve an issue that adversely affected the daily log rotation process for the Jetty request log files.
PF-16733	Lessen the lifetime of the inter-request state information to match its short-lived nature and thus reduced the memory footprint of PingFederate. (The setting is stored in the `size-limits.conf` file. For more information, see *Upgrade considerations* on page 683.)
PF-16683	Fixed an issue where users were not able to abort their requests for self-service password reset or continue with their requests under certain circumstances.
PF-16559	Resolved a rare upgrade issue where WS-Trust STS connections could be affected adversely.
PF-16514	Added a per-connection option to disregard a validation condition where 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. For more information, please refer to the `org.sourceid.saml20.util.VirtualIdentityUtil.xml` file.
PF-16475	Resolved an issue where configuration errors might occur to connections that are created with identical **Partner's Entity ID** value when letter case is ignore; for example, EXAMPLE.COM versus example.com.
PF-16455 and PF-16412	PingFederate is now more resilient when loading erroneous connections after an upgrade or during an import of a configuration archive.

Ticket ID	Description
	For instance, if multiple connections of the same role are using the same entity ID, PingFederate retains the latest connection automatically. Additionally, if PingFederate encounters incomplete connections, it ignores such entries without halting the rest of the process. Furthermore, such rare error conditions are now recorded in the server log, giving administrators the opportunity to review them at a later time.
PF-16436	Resolved a user interface issue in the administrative console where the console and the administrative API were configured to use native authentication and certificate authentication, respectively.

PingFederate 8.4.1 - August 2017

PingFederate 8.4.1 is a cumulative maintenance release for PingFederate 8.4, which introduced many new features, such as configuration at scale, OpenID Connect signed requests, JWT OAuth 2.0 authorization grants and client authentication, dynamic discovery of cluster nodes using AWS Roles, self-service account unlock, Docker support, and more. For a summary of the 8.4 release, see *PingFederate 8.4 - June 2017* on page 709.

Resolved issues

Ticket ID	Description
PF-16364	Resolved an issue where requests would fail if they invoked an authentication policy path that contained different instances of the Reference ID Adapter as back-to-back checkpoints.
PF-16329	The optional authentication sessions feature has been enhanced to handle scenarios where state servers may not always contain the same session-state information.
PF-16260	Resolved an issue where custom implementation of the OAuth client storage plug-in could fail after an upgrade.
PF-16190	When OAuth 2.0 Bearer Token is the chosen authentication scheme for SCIM outbound provisioning, PingFederate now transmits request parameters in the message body.
PF-16141	The OpenID Provider configuration endpoint (`/.well-known/openid-configuration`) now returns the request_object_signing_alg_values_supported value in the format of a regular string array.
PF-16119	Resolved a thread-safety issue in the Kerberos Adapter.
PF-16069	Improved the administrative console experience. Administrators are no longer required to clear the browser cache files in order for their browsers to render the user interface properly.
PF-15813	Resolved an issue where the administrative console did not allow an administrator to save an OAuth Assertion Grant IdP connection when WS-Trust was the only SP protocol that had been enabled (on the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen).
PF-15760	Upgraded to jose4j 0.5.7.
PF-15709	The SCIM inbound provisioning endpoints (`/pf-scim/v1/Users` and `/pf-scim/v1/Groups`) can now handle responses with over 10,000 records.
PF-15556	Resolved an issue where users of Internet Explorer were not able to fall through an instance of the Kerberos Adapter (or an instance of the IWA IdP Adapter that had been configured to allow Kerberos authentication only) to the next checkpoint in an authentication policy.

PingFederate 8.4 - June 2017

Enhancements

Configuration at scale

We have set a new bar for the number of IdP connections, SP connections, and OAuth clients that our customers can configure. We looked at the number of connections and OAuth clients that some of our largest customers were using and made improvements so that their investment could grow at least tenfold. This enhancement includes updates to paging, sorting, filtering, and searching in the administrative console to make it easier to find connections and OAuth clients. Configuration replication is also optimized to only replicate the connections that have changed, greatly reducing the amount of data that needs to be distributed and processed throughout a cluster when configuration changes.

Additionally, a new option is available to schedule the creation of backup configuration archives as an alternative to the generation of archives at sign-on to the administrative console. This option improves the responsiveness of the administrative-console sign-on experience for customers with many connections and OAuth clients because the backup process could take more than a few seconds.

The SDK support for custom storage of OAuth clients has been extended with a new interface, `ClientStorageManagerV2`. This interface includes a `search()` method, allowing customers to provide efficient implementations of the pagination and search functions exposed in the administrative console.

OpenID Connect signed requests

OpenID Connect defines a method to sign authentication requests sent from Relying Parties (RPs) so that the OpenID Provider (OP) can ensure that the request has not been tampered with in transit (see *https://openid.net/specs/openid-connect-core-1_0.html#JWTRequests*). We've added support for this in PingFederate, and this will open up new use cases. With signed requests, RPs can convey contextual information about the specific transaction the user is attempting. Authorization servers can then use that information to drive different interactions with the user, such as prompting the user for fine-grain consent. Customers can either implement a custom adapter to gain access to the data in the request parameter or look forward to future adapters shipped by Ping Identity that will expose this data. Further, we added support for both requiring signed requests from RPs and sending signed requests to OPs.

JWT OAuth 2.0 authorization grants

We have implemented support of the JWT profile for OAuth 2.0 authorization grants (see *https://tools.ietf.org/html/rfc7523#section-2.1*). The primary use case for this grant type is for cross-domain API transactions where the transaction is within the context of a user's consent but the user is not directly present. To use this grant type an OAuth client submits a signed JWT to an authorization server and receives an access token. This feature aligns with our existing support for the SAML assertion profile for OAuth 2.0 authorization grants.

To further support this use case we added a related feature to help customers that have OAuth clients that make cross-domain API calls and need to use this grant type with a remote AS. Clients that make calls to partner APIs that need to use a JWT authorization grant can exchange a local token for the necessary JWT using WS-Trust. This again aligns with our support of the equivalent SAML assertion profile.

JWT OAuth 2.0 client authentication

Another OAuth profile that is related to the JWT authorization grant, which is defined in the same specification, is OAuth 2.0 client authentication (see *https://tools.ietf.org/html/rfc7523#section-2.2*). We have also added support for this, and it allows OAuth clients to authenticate to an authorization server using a signed JWT. This provides a better security model than client ID and secret authentication as the private key used to sign the JWT remains secret to the client and does not need to be shared with an outside system. This support has been added for when PingFederate is the authorization server or when PingFederate is the OpenID Connect Relying Party.

Dynamic discovery of cluster nodes using AWS Roles

We've expanded our support for dynamic discovery (to support elastic scaling) of cluster nodes in AWS by leveraging the permissions of an AWS Role that's been assigned to an AMI. You can now use this option as an alternative to the existing option of setting an AWS IAM access key and secret for dynamic discovery. This new feature also adds the ability for PingFederate to discover nodes based on static and dynamic AWS AMI metadata through configuration of tags and filters.

Self-service account unlock

Adding to the existing self-service password reset feature of the HTML Form Adapter, we introduce a new feature to allow users to unlock accounts that have been locked due to password policy at the underlying LDAP directory. This optional feature can help in situations when users realize that they have mistyped their passwords several times and simply want to unlock their accounts, preserving their existing passwords.

Cluster-aware account lockout protection

Account lockout protection prevents user accounts from becoming locked at the underlying user repository based on too many failed authentication attempts. This protection, which is available 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 administrative console native authentication scheme), is now shared across nodes in a cluster. This enhancement helps in situations where PingFederate is deployed behind a load balancing infrastructure without sticky sessions.

Message customization and localization for password reset and account unlock

Administrators now have new options for customizing and localizing messages sent to users for self-service password reset and account unlock. Similar to the existing support that we already have for customizing email messages, SMS messages can also be customized. Furthermore, these messages can be localized based on the user’s preferred language, captured when the user initiates self-service password reset or account unlock.

Docker support

We continue to add features to help customers automate the deployment of PingFederate. With this release we now officially qualify on Docker.

Configurable cluster encryption strength

Cluster encryption strength is now configurable in PingFederate. The encryption strength varies depending on the cryptographic providers available from the Oracle Java SE Runtime Environment (Server JRE).

Security enhancement

The default TLS cipher suites for HTTPS listeners have been updated to follow the latest security best practices.

Other improvements

- Integration with Microsoft Office 365 has been improved to better support Microsoft Intune and Dynamics 365.
- PostgreSQL relational database support has been added and can be used for attribute lookup, OAuth persistent grant storage, OAuth client storage, logging, account linking, and outbound provisioning.
- Improved user provisioning efficiency to reduce execution time when multiple channels are configured.
- Tracking IDs are now included by default in audit log entries.
- Header based certificate authentication has been improved to be tolerant of missing PEM headers.
- The following bundled components and third-party dependencies have been updated:
  - OpenToken Adapter 2.5.6
  - PingID Integration Kit 1.4
  - Log4j 2 2.8.2
  - Jackson 2.7.8
  - Jetty 9.3.16
  - jose4j 0.5.5
  - Twilio Java SDK 7.4
  - UnboundID LDAP SDK for Java 3.2

Resolved issues

Ticket ID	Description
PF-15778	Fixed an access token validation issue where any claim with a data type other than string or list was not structured properly in the JSON response.
PF-15757	Resolved an issue where PingFederate did not update or clear the internal data store after changes had been made in the attribute mapping configuration for outbound provisioning.
PF-15756	The Kerberos Adapter now uses the correct Velocity template when a Kerberos Adapter instance is used as one of the checkpoints in an authentication policy and the authentication attempt has failed.
PF-15755	Authentication policies can now complete the failover process when an IdP adapter renders a Velocity template while returning a failure response status.
PF-15708	`EXCEPTION` messages from the `com.pingidentity.fsm.state.impl.PortalMenuState.getClientCountError` class are now categorized as `ERROR` (log level) in the server log.
PF-15630	Fixed an issue where the self-service password reset feature did not use information from the pf-accept-language cookie (if presented) to identify the locale of the user.
PF-15629	Resolved an OAuth client configuration issue where changes made to client authentication scheme were not saved occasionally.
PF-15564	Addressed an issue where contextual attribute values were not passed to the follow-on IdP connections when SP Authentication Policies were enabled
PF-15327	The `kerberos.error.template.html` template file now supports localization.
PF-15286	The `html.form.login.template.headerMessage` entry has been removed from the `pingfederate-message.properties` file as it is no longer applicable.
PF-15258	Resolved an issue where the username field values were always masked in the audit log for SLO requests through any OpenID Connect IdP connections.
PF-15201	Improved the validation process for the **Hostname(s)** field value in the LDAP data store configuration screen.
PF-15009	Improved the validation process for signed SAML 2.0 AuthnRequest messages; the AssertionConsumerServiceURL element value must be a full URL.
PF-14946	When processing SAML response messages that do not contain the Subject element, PingFederate now handles this error condition accordingly.
PF-14876	Updated the `log4j2.xml` configuration file with a new logger (`com.pingidentity.common.util.xml.XmlBeansUtil`). **Caution:** This new logger is disabled by default. When enabled, PingFederate may write sensitive user information to the server log. Consider enabling this logger for the sole purpose of troubleshooting in non-production environments only (and disabling this logger when it is no longer required).
PF-14602	Error conditions between the OAuth `/as/token.oauth2` endpoint and the clients are now better captured in the audit and server logs.
PF-14545 and PF-13334	When both signature verification and encryption are disabled, PingFederate now removes such partner certificates from the connection.
PF-14411	Fixed a dependency error when at least one attribute in an authentication policy contract deployed in an authentication policy was fulfilled by an OGNL expression.

Ticket ID	Description
PF-14361	Administrators must now select at least one virtual server ID when the **Restrict Virtual Server IDs** check box is selected on the **Virtual Server IDs** screen.
PF-14041	Improved the outbound provisioning process to handle a scenario where the **Timestamp Attribute** of an object does not contain any value.
PF-13733	Account lockout protection (via the Account Locking Service) has been extended to the Username Token Processor (UTP). PingFederate now rejects authentication attempts for a configurable amount of time after three failures. As needed, administrators may also override the failure threshold value per UTP instance.
PF-13698	When updating a SAML 2.0 connection by metadata, if encryption has not been enabled prior to the update, PingFederate now ignores the encryption certificate if any is included in the metadata.

PingFederate 8.3.3 - May 2017

PingFederate 8.3.3 is a cumulative maintenance release for PingFederate 8.3, which introduced many new features, such as self-service password reset (SSPR), hardware security module (HSM) hybrid mode, access token and authentication session sync, and improvements in OpenID Connect Relying Party support and OAuth authorization server. For more information, see the *release notes for PingFederate 8.3*.

Resolved issues

Ticket ID	Description
PF-15695	The CIDR Authentication Selector no longer returns an error when processing a request in which the source IP address includes a port number.
PF-15682	Improved self-service password reset (SSPR) for the release of PingID® authentication policies.
PF-15628	Updated the configuration file for the PingFederate Windows service to improve the responsiveness of the logging service.
PF-15581	Resolved an issue where a memory leak could occur when IdP connections were configured with SLO profiles or authentication sessions were enabled.
PF-15571	The Account Locking Service (for account lockout protection) now tracks the number of sign-on failures when resource owner credentials are validated against multiple LDAP Username Password Credential Validator mappings on the **OAuth Server** > **Resource Owner Credentials Mappings** screen.
PF-15472	Fixed a validation error where the `/authenticationPolicies/default` administrative API endpoint would reject a policy update that contained a reference to an IdP connection.
PF-15444	The `/pf-scim/v1/Users` and `/pf-scim/v1/Groups` SCIM inbound provisioning endpoints now return a 404 HTTP status code when the specified resource (a user or a group) is not found. (For more information, see the SCIM specification at *www.simplecloud.info/specs/draft-scim-api-01.html#anchor6*.)
PF-15205	When using PingID for SSPR, PingFederate now uses the regional information found in the `pingid.properties` file.
PF-15204	Resolved an issue where PingFederate could not load connections with partner's certificates after it was integrated with a Gemalto SafeNet Luna SA HSM.
PF-14956	Resolved an issue where PingFederate could not process encrypted SAML messages when it was integrated with an nShield Connect HSM.

PingFederate 8.3.2 - March 2017

PingFederate 8.3.2 is a cumulative maintenance release for PingFederate 8.3, which introduced many new features, such as self-service password reset (SSPR), hardware security module (HSM) hybrid mode, access token and authentication session sync, and improvements in OpenID Connect Relying Party support and OAuth authorization server. For more information, see the *release notes for PingFederate 8.3*.

Resolved issues

Ticket ID	Description
PF-15233 and PF-15208	Addressed two potential security vulnerabilities when using PingFederate with PingDirectory™, formerly known as UnboundID Data Store (SECNT007).
PF-15120	PingFederate installer for Windows now detects the installation directory of Oracle Server JRE 8 update 121.
PF-14996	When the **Change Password Email Notification** option was enabled on an HTML Form Adapter instance, PingFederate sent the change password notifications to users at their email address stored in the mail attribute regardless of whether another attribute was configured in the corresponding LDAP Username Password Credential Validator instance.
PF-14976	Fixed an issue where PingFederate did not write `AUTHN_ATTEMPT` events to the security audit log stored on an Oracle MySQL database.
PF-14972	Enhanced replay prevention on the resume URL when the authentication process involves user consent.
PF-14896	Resolved an issue where PingFederate did not update the **Issuer** field value in an IdP connection after loading the metadata from the OpenID Connect Provider.
PF-14877	Fixed an administrative console issue where administrators may see **OpenID Connect** as an available option on the **Connection Type** screen when creating an SP connection.
PF-14843	Self-service password reset (if configured) now works immediately after LDAPS is enabled on the underlying data store.
PF-14834	Fixed an issue where PingFederate could not process SAML authentication request messages that contain the Scoping element without the IDPList child element.

PingFederate 8.3.1 - January 2017

PingFederate 8.3.1 is a cumulative maintenance release for PingFederate 8.3, which introduced many new features, such as self-service password reset (SSPR), hardware security module (HSM) hybrid mode, access token and authentication session sync, and improvements in OpenID Connect Relying Party support and OAuth authorization server. For more information, see the *release notes for PingFederate 8.3*.

Resolved issues

Ticket ID	Description
PF-14819	New passwords submitted through self-service password reset are now only validated against the password policy of the associated data store, not the PingFederate internal service credentials password policy.
PF-14780	The outbound provisioning CLI (`provmgr.bat`) failed to start in a Microsoft Windows environment.

Ticket ID	**Description**
PF-14778	Resolved an issue where the administrative API could not create or update a connection with OGNL expressions.
PF-14771	PingFederate can now connect to the JSON Web Key Set endpoint (/pa/oidc/JWKS) on PingAccess.
PF-14631	Fixed an issue with the handling of client certificates that were set in HTTP headers by an HTTPS-terminating front end proxy server (SECNT005).
PF-14627	Upgraded to jose4j 0.5.3. PingFederate can now issue signed and encrypted JWT-based access tokens in an environment where PingFederate is integrated with a Thales HSM.
PF-14567	For any SLO-enabled SAML 2.0 IdP connection with multiple virtual server IDs, PingFederate now sends one logout request to the associated IdP when it receives a logout request at its SLO application endpoint (/sp/startSLO.ping).
PF-14423	Resolved an issue where the OAuth introspection endpoint (/as/introspect.oauth2) could not process validation requests when persistent grants were stored on an LDAP server.
PF-14372	PingFederate now includes SLO endpoint information in its SAML metadata.
PF-13996	Fixed an issue where PingFederate did not URL-encode the percentage character (if any was used in the **Partner's Entity ID** field of a connection) when constructing the **SSO Application Endpoint** URL on the connection summary screen.
PF-13624	When using OGNL expressions to define issuance criteria for authentication policy contract mapping, administrators may now reference attributes from an IdP connection using its **Partner's Entity ID** value; for example:

```
...
#idpCxnUsername = #this.get("idp.https://
sso.idp.local.SAML_SUBJECT")
...
```

In this example, the **Partner's Entity ID** value of the IdP connection is `https://sso.idp.local` and the referenced attribute is SAML_SUBJECT.

PingFederate 8.3 - December 2016

Enhancements

Self-service password reset

PingFederate now offers self-service password reset (SSPR) for end users to recover their accounts in the event of forgotten passwords. Integrated into the HTML Form Adapter and Password Credential Validator (PCV) framework, end users can now reset their passwords via one of four different mechanisms:

- One-time link via email
- One-time password via email
- One-time password via text message
- PingID®

The LDAP Username PCV supports the SSPR features, relying on additional configuration to look up the required attributes for the chosen reset mechanism. Custom PCV implementations may also be developed to offer SSPR features for users stored in non-LDAP data sources.

Hardware security module hybrid mode

Configuring PingFederate to use a hardware security module (HSM) is now simpler with a new hybrid mode option. HSM hybrid mode allows cryptographic material to be stored either locally on the file system or on an attached HSM (such as SafeNet or Thales). At creation or import time, administrators can choose where to store the certificate and key material. This feature can be used as part of a transition strategy to an HSM-only configuration for optimal security in your deployment.

Access token and authentication session sync

To provide end users with a consistent experience across web and API based native applications and single-page applications (SPAs), the validation of OAuth access tokens and authentication sessions can now be linked together.

Enabled through a policy on Access Token Manager instances, the validation of the access token can depend on an associated authentication session from which it was authorized. Activity on the access token, triggered during a validation or introspection call to PingFederate, can subsequently update the activity on the authentication session thereby centrally managing idle and maximum lifetime session controls. Additionally, should an end user sign off and revoke the associated authentication session, access tokens that rely on it can be considered invalid.

This capability can be used in conjunction with PingAccess® to support its authorization controls on both web and API resources.

OpenID Connect Relying Party improvements

Relying Party support has been enhanced for improved interoperability and ease of administration.

Form POST and **Form POST with Access Token** login types have been added to complement the existing Authorization Code Flow support to provide a more optimized login process.

Custom request parameters can now be added to provide additional partner specific parameters during login. As an example, this may be used for integrating Google login services to provide its hd parameter (see *https://developers.google.com/identity/protocols/OpenIDConnect#hd-param*).

Third party initiated login is now supported as defined in the OpenID Connect standard (see *openid.net/specs/openid-connect-core-1_0.html#ThirdPartyInitiatedLogin*).

Additionally, the administrative API has been extended to include the management of OpenID Connect Browser SSO IdP connections.

OAuth authorization server enhancements

OAuth authorization server capabilities have been enhanced for greater deployment flexibility and reduced administrative overhead.

The storage of clients can now be customized using a new `ClientStorageManager` interface available in the PingFederate SDK. This provides another deployment option for managing clients in addition to the existing options of local XML configuration files and a relational database system.

Access token validation has been simplified to ease application integration. Resource server clients can be configured to allow validation to occur against all eligible Access Token Manager instances automatically. When enabled, resource server clients no longer need to specify additional parameters (such as access_token_manager_id and aud) to disambiguate the Access Token Manager instance. This feature can also simplify interactions with PingAccess by avoiding the need to align resource URIs between it and PingFederate.

Registered claims (such as exp) within JSON Web Tokens formatted access tokens can now be overridden within attribute fulfillment. If the aud claim must be fulfilled at runtime, instead of relying on the static **Audience Claim Value** configuration setting, the aud will be ignored during access token validation.

Other improvements

The following bundled components and third-party dependencies have been updated:

- Jetty 9.3.14 (SSLv2Hello disabled)
- jose4j 0.5.2

Resolved issues

Ticket ID	Description
PF-14397	Upgraded to Jetty 9.3.14 to resolve a rare error condition (`java.lang.StackOverflowError`).
PF-14369	OAuth clients that authenticate through IdP connections no longer fail to do so when message customizations have been configured in such connections.
PF-14359	PingFederate now sets the Accept HTTP request header to `application/json` in its SCIM outbound requests to delete groups.
PF-14222	When an SP adapter instance was mapped to a URL on the **Target URL Mapping** screen, PingFederate always failed to locate the SP adapter instance if the adapter instance was overridden within an IdP connection.
PF-14178	The SSO Directory Service excluded OpenID Connect (OIDC) IdP connections from its responses.
PF-14172	The `/authenticationPolicies/default` administrative API endpoint failed to create or update a policy when the policy included at least one `attributeRules` with `"fallbackToSuccess": false`—the equivalent of creating or updating a policy in the administrative console with at least one rule, in which the **Default to Success** option was disabled; for example:

```
...
"attributeRules": {
  "items": [
    {
      ...
    }
  ],
  "fallbackToSuccess": false
}
...
```

Ticket ID	Description
PF-14147	When an administrator deletes one or more partner protocol endpoints from a connection, the administrative console now removes the intended entries.
PF-14141	OIDC Relying Party support has been enhanced to handle multivalued attributes.
PF-14113	Common Domain Service (for IdP Discovery) now works with the pf.runtime.context.path property in the `run.properties` file.
PF-14098	The `/idp/adapters` and `/idp/adapters/{id}` administrative API endpoints failed to create or update IdP adapter instance when the adapter instance included at least one `attributeSources` lookup with `"memberOfNestedGroup": true`—the equivalent of creating or updating an IdP adapter instance in the administrative console with at least one LDAP data store lookup to query the **memberOf** attribute with the **Nested Groups** option; for example:

```
...
"attributeMapping": {
  "attributeSources": [
    {
      "type": "LDAP",
      ...
      "memberOfNestedGroup": true
    }
  ],
```

Ticket ID	Description

	`   ...`
	`}`
	`...`

PF-14097	When loading metadata from an OIDC partner in an IdP connection, if the partner's HTTPS endpoint requires a certificate to be imported via the **Trusted CAs** screen, the certificate will now be trusted immediately after the import without requiring a restart.
PF-14083	SP-initiated SLO no longer fails if the pf.runtime.context.path property is set in the `run.properties` file.
PF-14029	The administrative console now prevents administrators from removing a JDBC data store that is used by one or more authentication policy contracts.
PF-13968	The PingFederate IdP server used to reject signed authentication requests when all the following conditions were met: • The **SP-initiated SSO** and **SP-initiated SLO** profiles were enabled. • The **Require digitally signed AuthN requests** option was disabled. • The SP used different signing certificates to sign its authentication requests and SLO messages. PingFederate now logs a warning message to the server log instead.
PF-13967	In a clustered PingFederate environment, an engine node now merges the new keys obtained from the console node during a configuration replication process into its `pf.jwk` file.
PF-13952	The Windows service file (`PingFederateService.conf`) has been improved to resolve a compatibility issue between the Thales nShield client driver and Oracle Java 8.
PF-13824	When PingFederate was configured with a managed SP connection to PingOne® and such connection was subsequently used as the source to create a new connection (via the **Copy** feature on the **SP Connections** screen), the connection to PingOne would break. As a result, users could not sign on to PingOne dock.
PF-13622	Improved the Composite Adapter to handle null attribute values in its attribute mapping process.
PF-13522	The HTML Form Adapter's default sign-on template file (`html.form.login.template.html`) now renders properly when invoked by Microsoft Office 365 applications.
PF-12734	The `/dataStores` administrative API endpoint does not require an `id` value anymore to create a new data store. A system-generated, read-only value will be assigned instead.
PF-12489	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.

PingFederate 8.2.2 - October 2016

PingFederate 8.2.2 is a cumulative maintenance release for PingFederate 8.2, which introduced many new features, such as OpenID Connect Relying Party support, authentication session caching, and password management improvements. For more information, see the *release notes for PingFederate 8.2*.

Resolved issues

Ticket ID	Description
PF-14033	Increased the maximum allowed size for an HTTP request header to handle requests with large headers (for example, a request with a Kerberos ticket whose user belongs to many groups).
PF-14017	Addressed an issue where the LDAP Username Password Credential Validator requested an LDAP server to sort a result set with one result, which could inadvertently cause a delay in receiving a response from the LDAP server.
PF-13994	Improved the responsiveness of the administrative console when storing OAuth clients in database.
PF-13982	Resolved an issue where the Kerberos Adapter and the Kerberos Token Processor could not extract the SID attribute value from Kerberos tickets.

PingFederate 8.2.1 - September 2016

PingFederate 8.2.1 is a cumulative maintenance release for PingFederate 8.2, which introduced many new features, such as OpenID Connect Relying Party support, authentication session caching, and password management improvements. For more information, see the *release notes for PingFederate 8.2*.

Resolved issues

Ticket ID	Description
PF-13950	PingFederate now redirects requests properly based on the **HTTP Header for Hostname** setting on the **Server Configuration** > **Server Settings** > **System Options** screen.
PF-13948	Resolved an issue where PingFederate finished its startup process without starting the integrated RADIUS server (part of the bundled PingID PCV).
PF-13928	PingFederate only returned a partial list of attributes for a selected object.
PF-13905	Resolved a logging issue that could lead to excessive memory usage.
PF-13837	Resolved an issue where attribute values were not consistently masked in the server log despite a proper configuration.
PF-13836	The upgrade process no longer copies older program files (from the previous releases of the PingID integration kit) to the new PingFederate 8.2.1 installation.
PF-13835	Improved the account linking process to handle the creation of new users without generating an `Unexpected exception` error condition. Note that new users were created despite the error message.
PF-13829	Improved PingFederate to validate OCSP responses signed using RSA SHA256, SHA384, or SHA512.
PF-13828	Enhanced the administrative console to handle the scenario where it cannot establish a JDBC connection with the database server that has been configured to store OAuth clients.
PF-13822	Administrative users without the **User Admin** role can now change their own passwords on the **Server Configuration** > **Account Management** screen.
PF-13740	Reverted the name of a parameter for the `/serverSettings/notifications` administrative API endpoint to notifyAdminUserPasswordChanges. Administrative API calls written for PingFederate 8.1.4 (and earlier versions) are now compatible with version 8.2.1.

Ticket ID	Description
PF-13730	Resolved an issue where PingFederate did not include the event type (`OAuth`), the subject, and the protocol (`OIDC`) in the audit information for requests sent to the UserInfo endpoint.
PF-13729	When IdP connections were used as part of the authentication policies, PingFederate failed to write information to the audit log after the users had finished with such IdP connections.
PF-13706	Updated the bundled PingID Integration Kit to version 1.3.3.
PF-13670	When validating redirection URLs, the matching algorithm has been improved when evaluating the entries defined on the **Server Configuration** > **Redirect Validation** screen. A more specific match is considered a better match and an exact match is considered the best match.

PingFederate 8.2 - August 2016

Enhancements

OpenID Connect Relying Party support

The OpenID Connect protocol is now available for browser-based SSO when PingFederate is acting as a service provider - a *Relying Party* in OpenID Connect terminology. This support complements the pre-existing OpenID Provider (OP) functionality and enables customers to make greater use of this modern protocol for partner use cases.

OpenID Connect can be configured using IdP connections and mapped into last mile integration options (such as OpenToken or Agentless Integration Kits) to support SSO into existing applications. These connections can also be bridged to other SSO protocols to act as a federation hub.

Authentication session caching

PingFederate now offers the ability to cache authentication sessions, giving administrators control over how often end users need to reauthenticate with any IdP adapter or IdP partner. Authentication session policies can be defined both globally and for individual authentication sources.

This feature can also be used to create a centralized session management experience for end users accessing protected applications through the Ping Identity Federated Access Management solution. PingAccess® web sessions will synchronize the session’s active state at PingFederate via calls to the OpenID Connect Session Revocation API.

Authentication policy contract mapping for OAuth

Authentication policy contract support has been added to OAuth and OpenID Provider configuration, enabling administrators to centrally define authentication logic and attribute mapping that can be reused across many varying partners and protocols.

As part of the Federated Access Management solution, these capabilities enhance the contextual access management capabilities when deployed alongside PingAccess and PingID®.

Password management improvements

New password management capabilities have been added to improve usability and end-user security.

Optionally enabled in the HTML Form Adapter, end users can be warned during login about an upcoming password expiry. This allows end users to better manage when and where they perform a password update to balance security policy compliance with user experience.

Additionally, when end users update their password through the HTML Form Adapter, they can be emailed a notification about this event. End users suspicious about this event can then take action to ensure their account remains secure.

Improved SLO support

To improve security and the end-user's experience, Single Logout (SLO) support has been improved to cover more use cases and protocol mappings. When SLO is initiated, PingFederate sends SLO requests to the applicable partners associated with the user's session and clean up any local sessions.

The following configuration models now support SLO:

- Federation hub
- Adapter-to-adapter mappings
- OAuth and OpenID Connect mappings using Browser SSO IdP connections

Authentication context overrides

When interacting with an IdP partner, PingFederate now offers improved control over how authentication contexts are communicated with it. Available as an override on IdP connections, local authentication context values can be mapped into remote values to dictate how they are translated. This provides greater flexibility when working with IdP partners that have fixed names for authentication mechanisms that may differ from your desired local representation.

OAuth enhancements

OAuth authorization server capabilities have been enhanced with the following features:

- Support for OAuth 2.0 Token Introspection (as defined in *RFC7662*) has been added to provide a standards-based mechanism to validate tokens.
- Encryption can now be configured on JSON Web Token (JWT) formatted OAuth access tokens.
- The error responses for the Resource Owner Credentials grant type now provide more meaningful error_description parameter details. Messages can additionally be localized and customized through the PingFederate localization framework.
- OAuth Persistent Grant Expiry can now be defined in minutes.
- Elliptic Curve Digital Signature Algorithm (ECDSA) is now available for JWT formatted access tokens.
- The JSON Web Token Access Manager can now be configured to publish a JSON Web Key Set with the keys/certificates that can be used for signature verification of access tokens.

Administrative API enhancements

The administrative API has been extended to include the following workflows:

- Authentication selectors
- Authentication policies
- OAuth attribute mapping in IdP connections
- SAML 2.0 Bearer OAuth grant type mapping

Other improvements

- Inbound provisioning has been enhanced to support sorting and pagination of resources as defined in the SCIM 1.1 standard.
- Audit logging enhancements to provide better visibility into failed authentication attempts.
- The HTTP Request Parameter Selector now supports wildcard based matching.
- The license update process is now simpler and can be applied to a cluster during configuration replication.
- Performance improvements to both OpenID Connect Policy administration and OAuth flows.
- The Log4j 2 configuration has been optimized for performance by defaulting to INFO level and disabling console logging.
- The Active Directory and Kerberos configuration now supports the definition of fully qualified domain names for key distribution centers.
- The domain of the PF cookie is now configurable to facilitate sharing of the user's session amongst engine nodes using differing hostnames under a common domain.

- Email notification templates now support HTML formatting.
- The default user-facing screens have been redesigned.
- Various administrative console improvements.
- The PingID Integration Kit 1.3.1 is now bundled with PingFederate.
- The following bundled components and third-party dependencies have been updated:
  - OpenToken Adapter 2.5.5
  - Log4j 2 2.6.1

Resolved issues

Ticket ID	Description
PF-13588	The soLingerTime property of the secondary HTTPS connector (httpsSecondaryConnector) now uses the correct property name (http.soLingerTime) in the `<pf_install>/pingfederate/etc/jetty-runtime.xml` file.
PF-13585	PingFederate now verifies database connections if a SQL statement (the **Validate Connection SQL** field value) is provided in the JDBC data store configuration.
PF-13336	The **Attribute Sources & User Lookup** configuration does not report an error anymore when the tokenGroups AD user attribute is configured as a binary attribute in the LDAP data store configuration.
PF-13335	The OAuth token endpoint (`/as/token.oauth2`) has been enhanced to return an HTTP status code of 500 when a token request is interrupted by an unexpected error, such as a data source failure or a runtime exception.
PF-13202	Resolved an issue where a request did not failover gracefully from the Kerberos Adapter through an authentication policy.
PF-13048	The attempts to change password (through the HTML Form Adapter) from locked out users no longer trigger PingFederate to write the **Unexpected Runtime Authn Adapter Integration Problem** error message in the server log. Note that the user experience and the audit log behavior remain the same: • The attempts are rejected with this message: Your account is locked. Please contact your system administrator. • The attempts are recorded to the audit log.
PF-12906	Improved a shell script (`<pf_install>/pingfederate/sbin/pingfederate-run.sh`) that the PingFederate install script uses.
PF-12751	When creating or updating a connection, the administrative API no longer requires the validFrom and expires properties of a certificate to be accurate down to the millisecond. Both of the following snippets are now accepted: • Without milliseconds: `... "validFrom":"2013-05-16T10:35:39Z", "expires":"2056-05-10T10:35:39Z", ...` • With milliseconds: `... "validFrom":"2013-05-16T10:35:39.000Z", "expires":"2056-05-10T10:35:39.000Z",`

Ticket ID	Description
	`...`
PF-12709	PingFederate no longer writes an `ERROR` message with a stack trace when a request results in a failure through an authentication policy. The simplified information is categorized as `DEBUG` information now.
PF-12648	The **Server Configuration** > **Account Management** screen is only accessible when one of the following conditions is met: • The administrative console is configured to use native authentication. • The administrative API is configured to use native authentication. • Both the administrative console and the administrative API are configured to use native authentication (the default as specified in the `<pf_install>/pingfederate/bin/run.properties` file).
PF-12378	The **Server Configuration** > **Certificate Management** configuration no longer results in an error when PingFederate encounters a connectivity issue with the OAuth client data store.
PF-12364	PingFederate now properly handles the error condition where the wa=wsignoutcleanup1.0 query parameter is incorrectly sent to the `/idp/prp.wsf` endpoint.
PF-12325	Resolved an issue where OAuth parameters client_id and client_secret were accepted as URL query parameters.
PF-12274	Improved the `<pf_install>/pingfederate/server/default/conf/terse.example.log4j2.xml` file; for example, audit information is now preserved.
PF-12136	Resolved an issue where the Kerberos Adapter and the Kerberos Token Processor could not extract the SID attribute value when the Kerberos environment was configured to use AES encryption.
PF-12131	PingFederate does not write an `ERROR` message with a stack trace anymore when the Kerberos Adapter receives an NTLM token (as opposed to a valid Kerberos ticket). The simplified information is categorized as `DEBUG` information now.
PF-11920	When issuing JWT formatted OAuth access tokens based on refresh tokens, PingFederate did not include the client_id information in the access tokens.
PF-11411	Resolved an issue where the outbound provisioning threads were closing after 60 seconds.
PF-11209	When a WS-Trust STS transaction completes successfully using the Kerberos Token Processor, PingFederate now writes the user identifier to the audit log.

PingFederate 8.1.4 - July 2016

PingFederate 8.1.4 is a cumulative maintenance release for PingFederate 8.1, which introduced many new features, such as advanced authentication policies, elastic scaling compatibility, and metadata publishing and consuming. For more information, see the *release notes for PingFederate 8.1*.

Resolved issues

Ticket ID	Description
PF-13353	The **Crypto** administrative role is no longer required to create a connection; only the **Admin** administrative role is required.

Ticket ID	Description
PF-13271	Deploying a configuration archive file to the `<pf_install>/pingfederate/server/default/data/drop-in-deployer` directory no longer triggers a certificate error that may block access to the PingFederate administrative console via LDAP authentication.
PF-13215	Resolved a compatibility issue in the LDIF scripts designed for UnboundID Data Store. The new scripts can be found in the `<pf_install>/pingfederate/server/default/conf/access-grant/ldif-scripts` directory.
PF-12787	Resolved an issue where PingFederate might stop responding after a non-blocking socket operation could not be completed.
PF-12720	In rare occasions, PingFederate installer for Windows might fail to detect the progress of the upgrade and report the upgrade as succeeded.
PF-12683	The PingFederate install script now offers the opportunity to update PingFederate's JAVA_HOME environment when the user's JAVA_HOME environment points to a different location.
PF-12682	PingFederate omitted the CN of expiring certificates in server log messages.
PF-12491	PingFederate ignored the default target URL of an adapter-to-adapter mapping when a request was initiated with both the IdpAdapterId and SpSessionAuthnAdapterId parameters and no default SSO URL was specified on the **Service Provider** > **Default URLs** screen.
PF-12481	An end user might encounter a redirect loop after initiating an SLO request.
PF-12469	Improved the `hsmpass` and `obfuscate` command-line utilities to handle hotfix packages.
PF-11683	Improved the redirect validation process for the InErrorResource parameter.

PingFederate 8.1.3 - May 2016

PingFederate 8.1.3 is a cumulative maintenance release for PingFederate 8.1, which introduced many new features, such as advanced authentication policies, elastic scaling compatibility, and metadata publishing and consuming. For more information, see the *release notes for PingFederate 8.1*.

Resolved issues

Ticket ID	Description
PF-12473	Editing the configuration of a token translator instance that is currently in-use by one or more WS-Trust requests no longer results in a deadlock.
PF-12468	Improved the handling of simultaneous requests from the same origin.
PF-12457 and PF-12454	Resolved an access issue in the `/ConfigArchive` administrative API endpoint and the **Server Configuration** > **Configuration Archive** screen in the administrative console.
PF-12450	The administrator audit log (`admin.log`) has been enhanced to include activities from configuration archive import.
PF-12412	Improved the Kerberos Adapter to resolve a browser looping issue when the following conditions were met: • PingFederate engine nodes (in a clustered PingFederate environment) were deployed behind a load balancer or a reverse proxy server without sticky sessions.

Ticket ID	Description
	• The authentication flow involved an instance of the Kerberos Adapter that was part of an instance of the Composite Adapter.
PF-12392	OAuth persistent grants that are stored in an LDAP directory server are now removed as the grants expire.
PF-12375	Custom data store filters could not be modified.
PF-12300	In a clustered PingFederate environment, the `/pf/JWKS` endpoint from the engine nodes might serve different keys for the same key ID (`kid`).
PF-12135 and PF-12134	Enhanced PingFederate to support surrogate pair characters in the administrative console and for runtime events.

PingFederate 8.1.2 - April 2016

PingFederate 8.1.2 is a cumulative maintenance release for PingFederate 8.1, which introduced many new features, such as advanced authentication policies, elastic scaling compatibility, and metadata publishing and consuming. For more information, see the *release notes for PingFederate 8.1*.

Resolved issues

Ticket ID	Description
PF-12383	Resolved a logging issue that could lead to degraded performance when an error had occurred.
PF-12323	Extended contract is now available for all IdP adapters.
PF-12322	When the **Adapter-to-Adapter Mappings** configuration contains one or more mappings from only one IdP adapter instance, PingFederate no longer returns a *Sign On Error* message when an adapter-to-adapter request provides an IdpAdapterId query parameter without a value.
PF-12289	Resolved a security issue in the administrative console.
PF-12285	PingFederate did not issue OAuth refresh tokens when the **Roll Refresh Token Values (default policy)** check box was selected and the **Minimum Interval to Roll Refresh Tokens (hours)** field was set to `0` in the **OAuth Server** > **Authorization Server Settings** screen.
PF-12283	The LDAP query process has been improved to handle result sets that may contain a large number of objects.
PF-12238	Resolved a logging issue when PingFederate was deployed with Box Connector 1.0.
PF-12203	The Asynchronous Front-Channel Logout process has been enhanced to terminate sessions at the grant-management endpoint (`/as/oauth_access_grants.ping`).
PF-12182	PingFederate now supports *Proof Key for Code Exchange by OAuth Public Clients* (tools.ietf.org/html/rfc7636) through a new parameter (code_challenge_method) at the OAuth authorization endpoint (`/as/authorization.oauth2`).
PF-12158	Resolved an issue where LDAP schema caching might lead to a high usage of memory.

PingFederate 8.1.1 - February 2016

PingFederate 8.1.1 is a cumulative maintenance release for PingFederate 8.1, which introduced many new features, such as advanced authentication policies, elastic scaling compatibility, and metadata publishing and consuming. For more information, see the *release notes for PingFederate 8.1*.

Resolved issues

Ticket ID	Description
PF-12237	The Kerberos Token Adapter and Kerberos Token Processor now return a null value for the SID attribute when such attribute cannot be decrypted.
PF-12235	The policy engine now skips authentication policies that only contain closed-ended failed paths when processing a request.
PF-12190	The PingFederate administrative console no longer allows IdP connections that have been placed in one or more authentication policies to be deleted from the **IdP Connections** screen. For each of these in-use IdP connections, administrators can click **Check Usage** to determine which policy (or policies) must be reconfigured before the IdP connection can be removed.
PF-12142	In the **OAuth Server** > **Client Management** screen, pressing the Enter key in the search field no longer returns the administrator to the **OAuth Server** menu.
PF-12141	The **OAuth Server** > **Authorization Server Settings** screen now displays scope and scope group values in alphabetical order.
PF-12133	The optional **Enable 'Remember My Username'** feature in the HTML Form Adapter now works with the latest Google Chrome (version 48).
PF-12080	The maximum lifetime value of an access token in JSON Web Token format was incorrect.
PF-11601	Updated the time stamp format in `log4j2.xml` file for better compatibility with Oracle Database.
PF-11444	PingFederate now provides more information in the server log when the SCIM inbound provisioner fails to provision users to Active Directory.

PingFederate 8.1 - January 2016

Enhancements

Advanced authentication policies

PingFederate now offers the ability to define central authentication policies that chain together selectors, adapters, and IdP connections to control how end users login for SSO and OAuth use cases. Additionally, data source lookups can now be performed within IdP adapter instances to associate them more closely to the authentication source and make policy decisions based on their results.

As part of the Federated Access Management solution, these capabilities support context-sensitive adaptive authentication when combined with PingAccess and PingID®. Identity data following a first-factor of authentication can be augmented to make intelligent decisions on how to further authenticate a user. A common scenario for this could be group-based selection of multifactor authentication.

Metadata publishing and consuming

Browser SSO connection creation and ongoing administration is now simpler with SAML metadata publishing and consuming. This feature provides support for metadata to be retrieved from and published to HTTP or HTTPS URLs and compatibility with trust networks such as InCommon.

Partner connections can be associated with metadata URLs, and a policy provides controls over periodic checks of the metadata to automatically update certificates and contact information. If other connection changes are detected within the metadata, the PingFederate administrators can be notified via email of said changes to review for manual reconciliation. Additionally, certificate and key rotation features enable automated rollover of SAML signing and encryption certificates for partners that can consume metadata URLs.

Elastic scaling compatibility

PingFederate now provides improved compatibility with auto (or "elastic") scaling features provided by cloud infrastructure services including Amazon Web Services and OpenStack. The dynamic discovery mechanism enables new members to join an existing cluster without configuring IP addresses ahead of time; it enables the horizontal scaling of a cluster without manual intervention when traffic volume dictates additional resources are required.

LDAP directory and custom storage for OAuth persistent grants

For OAuth deployments requiring persistent grants, this data can now be stored in an LDAP directory as an alternative to a relational database system. Consult *System requirements* in the *Get started with PingFederate* guide for the list of qualified directory server products for this feature.

Additionally, PingFederate now allows developers to build their own OAuth Access Grant Manager, providing more options (beyond an LDAP directory server or relational database system) for how their access grant can be persisted. The PingFederate SDK provides the `com.pingidentity.sdk.accessgrant.AccessGrantManager` interface that customers can implement. A sample implementation is located in the `<pf_install>/pingfederate/sdk/plugin-src/access-grant-example/java` directory.

Active Directory Authentication Mechanism Assurance support

The Kerberos Adapter has been extended to return the SID (security identifier) from the Kerberos ticket as part of its contract. This new capability enables the checking of the user's Authentication Mechanism Assurance to determine if the user logged in to the Windows desktop via username/password or a smart card. Authorization policies, either within PingFederate or the target application, can then dictate what the user is allowed to do.

PingFederate installer improvements

Windows installer and Red Hat Enterprise Linux install script can now detect that a previous release is present on the system that was also installed using an earlier release of the installer and automate the upgrade.

Administrative API enhancements

The administrative API has been extended to include the following workflows:

- SAML 2.0 Browser SSO connections using Artifact or SOAP bindings and SLO
- SAML 1.x connections
- WS-Federation connections
- Target URL mappings
- Metadata URLs management
- Key pair and certificate rotation for self-signed certificates
- Authentication policy contracts
- Authentication policy contract mapping in IdP and SP connections

Security enhancement

- XML encryption features within SAML Browser SSO connections have been extended to support a primary XML decryption key and secondary XML decryption key.

Other improvements

- The default user-facing screens have been beautifully redesigned. Previously modified screens can be preserved by copying the respective Velocity template files from the old installation to the new installation. For more information, see *Copy customized files or settings* on page 756 in the *PingFederate Upgrade Utility user guide*.
- Kerberos Token Processor is now bundled with PingFederate.
- CIDR Authentication Selector now supports IPv6 addresses.
- UnboundID Data Store is now qualified for various directory requiring features.

- Oracle database 12c is now qualified.
- OAuth client ID is now available as a variable in applicable end user facing Velocity templates.
- Connection Mapping Contract has been renamed to Authentication Policy Contract.
- Better administrator experience when data stores are unresponsive.
- Capability to map the SAML 2.0 Authenticating Authority attribute value to an SP adapter contract.
- Upgraded to JGroups 3.6.5 (and JGroup MERGE3).

Resolved issues

Ticket ID	Description
PF-12240	PingFederate did not capture the response size in the request log.
PF-11817	The PingFederate administrative APIs did not save the contract fulfillment mappings when creating an SP connection with a pseudonym identity mapping and additional attributes from multiple data stores using one mapping.
PF-11685	PingFederate logged the runtime port as `-1` in the transaction log.
PF-11684	When the **Enable 'Remember My Username'** and **Allow Password Changes** check boxes were selected, and a user had successfully authenticated and saved the username, another user could not sign in using the same HTML Form Adapter instance from the same browser if a password change was required as well.
PF-11672	PingFederate could not provision users with default groups to the PingOne directory.
PF-11541	The PingFederate administrative console reported an error if it was configured to use LDAP authentication without a prior login using the default native authentication.
PF-11528	IdPs, who were invited by the customers of PingOne SSO for SaaS Apps, could not establish an managed SP connection to PingOne using the **Connect to PingOne** configuration wizard.
PF-11510	Just-in-time (JIT) provisioning was not updating group membership information consistently.
PF-11469	Resolved SCIM inbound provisioning issues on Active Directory with partitions.
PF-11432	PingFederate logged non-fatal `Unexpected exception` messages to the transaction log when handling OpenID Connect requests.
PF-11359	The `configcopy` utility failed and returned a `No such operation 'saveServerFile'` message.
PF-11318	Added randomization support in the resume path for the Kerberos Adapter.
PF-11317	Resolved configuration issues in JDBC connection pooling.
PF-11175	Resolved an issue where PingFederate was unable to replicate provisioner configuration file when provisioner was running at the same time
PF-11130	Improved PingFederate Upgrade Utility to remind the administrators to migrate from an older version of Username Token Processor (if found) to the newer Username Token Processor bundled with PingFederate since version 7.2.
PF-11085	When the **Allow Password Changes** check box is selected, the HTML Form Adapter reauthenticates the user using the new password immediately after a successful password change request. PingFederate 8.1 introduced a new HTML Form Adapter setting, **Post-Password Change Re-Authentication Delay**, to handle the scenario where a time delay between the password change and the reauthentication attempt is required.

Ticket ID	Description
PF-11079	Previously configured OAuth access token mapping returned an error when the partner's entity ID of the corresponding IdP connection had changed.
PF-10882	Starting with PingFederate 8.1, you are not required to place an instance of the Requested AuthN Context Authentication Selector as the last checkpoint in a policy in order for the **AuthN Context Attribute** value to be mapped into an assertion. Similarly, you are also not required to place an instance of the CIDR Authentication Selector as the last checkpoint in a policy in order to use the **Result Attribute Name** value to fulfill an attribute contract or for issuance criteria.
PF-10811	When querying multiple data stores for contract fulfillment or token authorization, if a data store uses results from previous queries as input, and if the previous queries return no result, PingFederate should continue the SSO process by moving on to the next data store in the setup. For more information, see *Attribute mapping with multiple data sources* on page 603.
PF-10569	The metadata import process did not compare the bindings included in the metadata against those defined in the SAML specifications using a case-sensitive matching rule. As a result, requests might fail at runtime. This issue is now resolved; for example: • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST` is valid. • `urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Post` is invalid because the binding ends with `HTTP-Post` as opposed to `HTTP-POST`.
PF-10149	For email notification using SSL or TLS, hostname verification of the certificate is now available. 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 PingFederate Upgrade Utility preserves the configuration without activating the hostname verification option for compatibility reasons. Administrators should consider activating this new option for greater security.

PingFederate 8.0.4 - November 2015

PingFederate 8.0.4 resolved the following issues.

Ticket ID	Description
PF-11463	Resolved a potential security vulnerability.
PF-11413	The `/pf-admin-api/v1/idp/spConnections` administrative API no longer throws a `NullPointerException` error after upgrade.
PF-11328	Upgraded the `jaxb-impl.jar` file from version 2.1.7 to version 2.2.14 to resolve a duplicate Java class error when the WAM Integration Kit was deployed with the library files from Oracle (for the use case of WAM plug-in for Oracle Access Manager).
PF-11287	When importing a metadata file, PingFederate no longer throws a `NullPointerException` error when the metadata file is invalid.
PF-11198	Resolved a duplicate Java class error when the WAM Integration Kit was deployed with the `oraclepki.jar` file from Oracle (for the use case of WAM plug-in for Oracle Access Manager).
PF-11161	The OAuth authorization endpoint (`/as/authorization.oauth2`) now returns a 403 Forbidden response when it receives an HTTP HEAD request.
PF-11146	The LDAP connection pool does not leak connections anymore after its associated data store settings have been modified.

Ticket ID	Description
PF-11106	PingFederate returned a 400 Bad Request response when SCIM inbound provisioning was configured with one or more custom SCIM attributes and the **Nested Groups** check box was selected for the memberOf attribute in the configuration for SCIM responses.
PF-11082	The `/pf-scim/v1/Groups` endpoint for SCIM inbound provisioning did not return the schema attribute.
PF-10911	PingFederate did not remove an attribute from the access token mapping configuration after such attribute had been removed from the access token contract.
PF-8624	End-user credentials for all WS-Federation transactions should always be masked in the server log.

PingFederate 8.0.3 - September 2015

PingFederate 8.0.3 resolved the following issues.

Ticket ID	Description
PF-10975	SSO and SLO requests using the SOAP binding failed with Oracle Java 8 Update 51.
PF-10963	Installation of the PingFederate Windows service (via the `install-service.bat` file) failed on Oracle Java 8.
PF-10894	A race condition could occur in the authentication process through the integrated Kerberos Adapter during high request volume.
PF-10878	An obfuscated password for cluster authentication on an engine node inadvertently caused PingFederate failed to start after an upgrade.
PF-10858	Despite the fact that an SSO request completed successfully, PingFederate logged a failure in the audit log when the optional **Default Target URL** was omitted in an IdP connection that was deployed as part of the federation hub use case.

PF-10846	Improved the HTTP responses returned by the UserInfo endpoint as follows:
	• When the OpenID Connect role is not enabled, the UserInfo endpoint returns `404 Not Found`.
	• When the OpenID Connect role is enabled but the request has failed the Token Authorization framework (issuance criteria), the UserInfo endpoint returns `403 Forbidden` with an WWW-Authenticate header; the header value reads:
	`error_description="ACCESS_DENIED" realm="Userinfo" scope="<a list of applicable scopes>"`
	• When the OpenID Connect role is enabled but the request has failed the attribute mapping process, the UserInfo endpoint also returns `403 Forbidden` with an WWW-Authenticate header; the header value reads:
	`realm="Userinfo" scope="<a list of applicable scopes>"`

PF-10810	Resolved a security issue in the administrative console. For more information, visit the Ping Identity *Support* website and refer to the Security Advisory SECADV011.
PF-10797	Improved the sample code for password credential validator (`SamplePasswordCredentialValidator.java`) to illustrate the use case better.
PF-10757	Enhanced `obfuscate.bat` and `obfuscate.sh` to support special characters. (Administrators must enclose passwords with special characters by single quotation marks.)
PF-10711	Provided support for boolean and integer values in JWT access token manager.

Ticket ID	Description
PF-10430	The inbound provisioning process used an incorrect naming context to search users, which might return an error in some Active Directory environments.
PF-10329	The `exclude-patterns` sample under X-Frame-Options in the `<pf_install>/pingfederate/server/default/data/config-store/response-header-runtime-config.xml` file contained an incorrect sample.
PF-9839	When a signed AuthnRequest came with an AssertionConsumerServiceURL attribute, and the authentication failed, PingFederate sent the SAMLResponse (with a nonsuccess status) to the **Assertion Consumer Service URL** defined in the SP connection in error. PingFederate should send the nonsuccess status to the `AssertionConsumerServiceURL` value specified in the signed AuthnRequest.

PingFederate 8.0.2 - August 2015

PingFederate 8.0.2 resolved the following issues.

Ticket ID	Issue
PF-10657	The PingOne Directory password credential validator fails to authenticate when PingFederate is deployed behind a web proxy server.
PF-10646	The **Connect to PingOne** configuration wizard fails when PingFederate is deployed behind a web proxy server.
PF-10622	The PingFederate SP server does not provision attributes from SCIM Enterprise Schema Extension through the support of custom attributes in the SCIM inbound workflow.
PF-10616	Kerberos authentication fails unexpectedly after the initial login has succeeded.
PF-10553	Upgraded JGroups to version 3.6.4 to resolve a JGroups issue: *FD_SOCK/FD: members are not unsuspected* (JGRP-1922). For more information, see *https://issues.jboss.org/browse/JGRP-1922*.

PingFederate 8.0.1 - July 2015

PingFederate 8.0.1 resolved the following issues.

Ticket ID	Issue
PF-10551	The PingOne Directory password credential validator in PingFederate 8.0 does not support the change password feature for expired accounts. It should.
PF-10437	An upgraded PingFederate 8.0 may fail to start if a Jetty patch was applied to the older PingFederate installation.
PF-10388	The LDAP Username Password Credential Validator in PingFederate 8.0 does not include the DN attribute in its responses after the LDAP credentials have been validated. It should.

PingFederate 8.0 - June 2015

- PingOne integration improvements
- PingFederate installers for Microsoft Windows and Red Hat Enterprise Linux
- Administrative API enhancements; introduced the capability to authenticate by X.509 certificates, LDAP or RADIUS accounts and extended coverage to:
    - IdP and SP Default URLs configuration
    - Resource Owner Credentials Mappings (OAuth)
    - Kerberos Realms and Active Directory Domains configuration
    - Export/Import SP Connection metadata
- SCIM Provisioning enhancements to support:

    - Custom attributes for inbound and outbound provisioning
    - Filtering and querying on attributes for inbound provisioning
    - Nested group membership for Outbound Provisioning
- Update SAML connections using metadata XML files
- Browser SSO protocol customization
- JSON Web Token Support in WS-Federation SP connections
- HTTP Request Parameter Authentication Selector
- Kerberos IdP Adapter
- Logging enhancements:
    - Upgraded to Log4j 2
    - Introduced a new access token tracking ID
    - Introduced a new audit log for the administrative console with detailed information for each event
- Security enhancements
- Other enhancements:
    - Redesigned the administrative console experience
    - Enhanced compliance with OpenID Connect specification
    - Upgraded to Jetty 9.2.11
    - Removed the Oracle Java SE Development Kit (JDK) dependency in favor of the Oracle Java SE Runtime Environment (Server JRE)
    - Restructured context-sensitive help in the administrative console to hyperlink to online documentation

Resolved Issues

PingFederate 8.0 also resolved the following issues.

Ticket ID	Issue
PF-9916	Cluster nodes throwing RuntimeExceptions during RPC calls can fail operations for the whole cluster
PF-9905	The “Message” column in the server-log-sqlserver.sql script is not large enough
PF-9904	The “Description” column in the audit-log-sqlserver.sql script is not large enough
PF-9869	A challengeable PCV is not being re-challenged when it throws a PasswordCredentialChallengeException
PF-9838	An error occurs when setting the TargetResource to a URL that has multiple '#' chars
PF-9837	In an Office 365 deployment, Lync locks out an account if their password was changed in AD
PF-9733	LDAP's memberOf Nested Groups check box value is not saved when using memberOf in an OGNL expression
PF-9729	The Log4j 2 appender PatternLayoutForDB throws NPE for null messages (resolved by upgrading to Log4j 2)
PF-9565	Windows service installation script causes cosmetic error during auto-configuration of PingFederateService.conf
PF-9549	Unable to create an SpConnection with multiple adapter mappings via the administrative API
PF-9476	The location value for IdpAdapterRef is incorrect in the administrative API's SpConnection
PF-9441	SOAP Binding does not output the fault details from the partner
PF-9359	Refreshing an OAuth token when a data store is not available revokes the access grant
PF-9355	MySQL access grant script fails if the sql mode is set to NO_ZERO_DATE

Ticket ID	Issue
PF-9200	Japanese characters are not displayed correctly in http.error.page.template.html template
PF-9162	OAuth grant table gets queried even when refresh tokens/re-use of persistent grant are not being used
PF-9098	Username is not audited when using the pseudonym identity mapping
PF-9082	A connection's signing algorithm cannot be changed without a certificate change
PF-9057	LDAP attributes are not always masked in the log
PF-9043	Using Nested Groups does not escape special characters in the LDAP search filter
PF-9042	In an Office 365 deployment, Outlook locks out account if their password has been changed in AD
PF-9040	Connection Deployer does not validate binary attributes importing a connection
PF-9039	LDAP Username Password Credential Validator should not return all attributes for a DN query
PF-9004	"javax.servlet.http.NoBodyResponse cannot be cast to org.sourceid.servlet.HttpServletRespProxy" error when receiving HEAD requests
PF-8993	Overriding default trust manager should be logged at DEBUG
PF-8989	Creating OpenToken Adapter via the administrative API causes error at runtime
PF-8962	Deadlock can occur when using database logging with failover
PF-8683	RSTR is missing the RequestedAttachedReference for encrypted SAML tokens
PF-8575	Audit log shows 'success' when RST validation fails

PingFederate 7.3 - January 2015

- Federation Hub
- SampleAuthenticationSelector SDK Sample
- SP Connections API
- Group Support for Inbound Provisioning
- Support for wreply in WS-Federation SSO
- Improved Redirect Validation
- Improved the cleanup process of expired persistent grants; added an index (`EXPIRESIDX`) for the `expires` column in the pingfederate_access_grant database table
- Hostname verification for LDAPS data stores
- Issuer restriction in anchored trust model for back-channel authentication and XML signature verification
- Elliptic Curve algorithms for certificate creation and XML signatures
- RSA certificate creation enhanced to include RSA SHA-2 signing algorithms and 4096 bit keys
- Application Name and Application Icon URL for SP Connections and Adapter-to-Adapter mappings
- Customizable response from /pf/heartbeat.ping
- Customizable template for HTTP error pages
- Customizable favicon.ico
- LDAP Username Password Credential Validator can now be configured to return additional user attributes beyond the username and user DN
- PingOne Directory Password Credential Validator is now bundled as part of the standard PingFederate installation
- Key algorithm and key size are now displayed in certificate management pages and the certificate details popup
- HTML Form Adapter now supports a configurable maximum session lifetime
- Support for nested LDAP groups in Outbound Provisioning
- Support retrieval of nested LDAP groups in Attribute Contract Fulfillment
- Upgraded JGroups to version 3.5.1

- Upgraded Jetty to version 8.1.16
- Over 60 other product issues resolved

PingFederate 7.2 R2 - September 2014

- Multiple Access Token Management Plug-in Instances
- Improved Attribute Mapping for OAuth access-token contract
- Multiple data-source lookups for OAuth workflows
- OpenID Connect / OAuth 2.0 form POST response mode
- Support User Info in the OpenID Connect ID Token
- Administrative API enhancements:
  - Configuration archive management
  - IdP Connection Metadata Export/Import
  - Certificate Management of Trusted CAs, SSL Client Keys, and SSL Server Certificates
  - Data Sources
  - Password Credential Validators
  - Adapter-to-Adapter Mapping
- Administrative Console Enhancements
- Other Enhancements:
  - The OAuth client_id parameter is now available in the IdP Adapter SDK interface
  - Support SID encoding for binary LDAP attributes (enabling claims-based authentication to Microsoft Outlook Web Access)
  - Various security enhancements
  - Over 40 other product issues resolved

PingFederate 7.2.1 - August 2014

- Fixed an issue for Office 365 Outlook use cases where user accounts could be locked after changing their passwords in Active Directory.
- Fixed an issue where users would not be provisioned correctly under certain conditions.
- Fixed an issue that prevented additional "Actions" in adapter configuration from executing correctly.

PingFederate 7.2 - June 2014

- Multiple Virtual Server IDs
- OpenID Connect-based Centralized Session Management
- LDAP Enhancements
  - Improved performance on LDAP bind operations
  - Ability to control timeout on LDAP attribute lookups
  - Better connection cleanup when an idle LDAP connection is removed from the pool
- Administrative API Enhancements
  - IdP and SP Adapters
  - Access Token Management (OAuth)
- Custom HTTP Response Headers
- Improved Target Resource Validation
- Outbound Provisioning Monitoring
- New Username Token Processor
- Redesigned Default User-Facing Screens
- Other Enhancements
  - Support for Java 8
  - Support for OAuth Symmetric Proof of Possession for Code Extension
  - Upgraded JGroups to 3.4.4.Final

  - Various security enhancements
  - Over 50 other product issues resolved

PingFederate 7.1.4 - June 2014

- Addressed a potential CSRF issue with the PingFederate Administration console.
- Updated the default Multiple SSO in Progress template to escape additional input variables (`speed.bump.template.html`).
- Addressed an issue where Office 365 Outlook users can have their user accounts to be locked out after they change their Active Directory passwords.
- Corrected the Content-Type header for interoperability with Office 365 OneDrive.
- Made available the $HttpServletRequest object in the IdP logout template (`template.idp.logout.success.page.template.html`).
- Resolve an issue with body-less HTML Form POST messages sent by Internet Explorer when HTML Form based authentication follows an unsuccessful IWA authentication within a Composite Adapter.
- Made an improvement to correctly handle LDAP queries for Distinguished Names having forward slashes (/).

PingFederate 7.1 R3 - March 2014

- Administrative API Enhancements
  - Server Settings (OAuth Use Cases)
  - OAuth Settings
  - Authorization Server Settings
  - OpenID Connect Policy Management
  - Client Management
  - IdP Adapter Mapping
  - Access Token Mapping
- OAuth Enhancements
  - OAuth 2.0 Token Revocation (RFC7009); an index (`CLIENTIDIDX`) was added for the `client_id` column in the pingfederate_access_grant database table
  - Access Grant API
- RADIUS Support Extended
  - Challenge-Handshake Authentication Protocol (CHAP) Support
  - Vendor-Specific Attributes
  - NAS-IP-Address Passed in Access-Request
- Enhanced Support for Reverse Proxy Deployments
- Other Enhancements
  - Allow PingFederate's session cookie to be optionally configured to be persistent
  - Allow the Access Token Verification/Validation Grant Type to be used in combination with other grant types
  - Various security enhancements
  - Over 20 other product issues resolved

PingFederate 7.1.3 - February 2014

- Corrected potential security vulnerability.

PingFederate 7.1.2 - December 2013

- Corrected an issue with artifact binding in clustered deployments that resulted in sporadic failed transactions. This issue only applied to version 7.1.1.

PingFederate 7.1 R2 - December 2013

- Administrative API

- Tracking ID Enhancements
- Outbound Provisioning
    - Microsoft SQL Server Support added for Internal Provisioning Data Store
    - OAuth 2.0 Bearer Token Authorization for SCIM 1.1 Plugin
    - Ability to Define Deprovision Method for SCIM 1.1 Plugin
- OAuth and OpenID Connect Enhancements
    - Extended OAuth to allow multiple scopes to be grouped together and referenced as a single scope, enabling scope downgrade during token refresh
    - Allow administrators to define a maximum token lifetime for OAuth Access Tokens
    - OpenID Connect now supports requesting that the authorization server (AS) use specified authentication contexts when processing the authentication request
- Enhanced the PingFederate SP server to validate the Audience element in SAML assertions against the Entity ID defined in Server Settings and the Virtual Server IDs defined within the IdP connection
- Added TemplateRendererUtil class and template-render-adapter-example to PingFederate SDK
- Allow administrators to define a maximum token lifetime for OAuth Access Tokens
- Extended HTML Form Adapter to allow usernames to be stored and pre-populated in the login form after a user's first successful authentication
- OpenID Connect now supports requesting that the AS use specified authentication contexts when processing the authentication request
- Partner Entity ID (Connection ID) now available as input parameter to all Identity Store Provisioner requests
- Extended OAuth to allow multiple scopes to be grouped together and referenced as a single scope, allowing scope downgrade during token refresh
- Reintroduced Luna HSM Support
- Upgraded Jetty to 8.1.14
- Upgraded JGroups to 3.3.4.Final
- Various security enhancements
- Over 40 other product issues resolved

PingFederate 7.1.1 - November 2013

- Corrected potential security vulnerability
- Fixed XML namespace issue that resulted in failed SSO transactions at partners with hardcoded dependencies on the `samlp` prefix
- Resolved XML parsing issue that resulted in PingFederate rejecting incoming authentication requests
- Corrected logging issue that resulted in AJP-based transactions to fail

PingFederate 7.1 - August 2013

- Added support for outbound provisioning via PingOne
- Added identity store SDK for inbound provisioning
- Provided Remote Authentication Dial-In User Service (RADIUS) support for console sign on and password credential validation
- Added hierarchical (parent/child) configurations for plug-in instances
- Related plug-in instance override capability added for connections
- Enabled overriding the global Default Target URL for connections and adapter-to-adapter mapping
- Made STS client authentication details available for token authorization
- Extended JMX runtime monitoring support
- Enabled usage checking for certificates, adapters, token translators, credential validators and data stores
- Store OAuth client configuration as XML files (new default, database storage optional)
- Upgraded Jetty to 8.1.9
- Upgraded JGroups to 3.3.0.Final
- Various security enhancements

- Over 80 other product issues resolved

PingFederate 7.0.1 - May 2013

Addressed potential security vulnerabilities found since the PingFederate 7.0 initial, limited-availability release.

PingFederate 7.0 - April 2013

- Added support for System for Cross-domain Identity Management (SCIM)
  - Inbound Provisioning
  - Outbound Provisioning
- Initial Support for OpenID Connect
- New Context sources available in Token Authorization and Attribute Mapping workflows
  - HTTP Request
  - *Client IP* Request
- Administrative Console Enhancements
- Upgraded Jetty to 8.1.8
- Added support for the SAML AuthnRequest Scoping element Request (see *Administrator's Manual*: Configuring the Requested AuthN Context Selector)
- Created the Cluster Node Adapter Selector for use within Adapter Selection Request (see *Administrator's Manual*: Configuring the Cluster Node Selector)
- Allow PingFederate runtime context path to be customized Request
- Security Enhancements
- Over 50 product issues resolved

Versions prior to 7.0

PingFederate 6.11 - December 2012

- Added password update via HTML Form Adapter
- Enhanced IdP Adapter Selector functionality:
  - Decision Tree Processing (see *Administrator's Manual*: Mapping Selector Results to Adapter Instances)
  - Connection Set Selector (see *Administrator's Manual*: Configuring the Connection Set Selector)
  - HTTP Header Selector (see *Administrator's Manual*: Configuring the HTTP Header Selector)
  - OAuth Scope Selector (see *Administrator's Manual*: Configuring the OAuth Scope Selector)
- Added localization for user-facing Web pages
- Added capability of defining globally an HTTP Header containing client IP addresses (see *Administrator's Manual*: Defining an HTTP Header for Client IP Addresses)
- Added OAuth fine-grain scope handling and JWT access-token support
- Added Dynamic JVM Resource Allocation
- Upgraded JGroups to 3.3.0.Alpha1 (snapshot from 10/30/12)
- Extended the Consent Form template to include the $entityId parameter
- Extended the OAuth Grants Management template to show grant type, scope, and the date/time a grant was issued and updated
- Improved SLO handling in scenarios where a user performs multiple SSO requests
- Improved the ability for administrators to define validation rules for HTTP request parameters
- Improved JDBC data store attribute lookup functionality to allow for retrieval of multiple values from a single database column
- Made security enhancements
- Removed the requirement to include TokenProcessorId/TokenGeneratorId query parameters for STS requests to an IdP/SP connection when only a single instance of the token-processor/generator type is configured for the connection
- Enabled PingFederate to use Jetty's NIO (New I/O) backed SSL Connectors by default

- Enhanced Microsoft Office 365 support to allow for Kerberos interoperability with active clients built on top of Microsoft Online Services Sign-In Assistant
- Enhanced LDAP error code handling in the LDAP Username Password Credential Validator to enable more specific error messages to be provided to end users
- Enabled PingFederate to interoperate with Microsoft Dynamics CRM 2011

PingFederate 6.10.1 - January 2013

- Replaced OpenToken Adapter with version 2.5.1 to capture security enhancements
- Other changes to address potential security vulnerabilities

PingFederate 6.10 - September 2012

- Token Authorization
- STS token exchange mapping
- OAuth client mutual TLS authentication
- OAuth 2.0 final draft compliance

PingFederate 6.9 - June 2012

- Microsoft Office 365 interoperability
- STS transaction events logged to audit log
- Upgraded Jetty and removed underlying JBoss infrastructure

PingFederate 6.8 - April 2012

- Added centralized AD Domain/Kerberos Realm configuration
- Added OAuth Client Management REST API
- Added optional expiration of OAuth persistent grants
- Added multiple redirect URIs per OAuth client
- Added optional restricted scope subsets per OAuth client
- Added configurable consent page omission per OAuth client
- Added OAuth transaction events logged to audit log

PingFederate 6.7 - February 2012

- Added Splunk Application for PingFederate
- Improved administrative console navigation and save performance
- Added LDAP connection pooling options for LDAP data stores

PingFederate 6.6 - December 2011

- Added contextual IdP Adapter selection using Adapter Selectors
- Added ability to chain multiple IdP adapters together using the Composite Adapter
- Added the ability to use multiple IdP data stores for attribute retrieval and mapping into an IdP attribute contract
- Added an HTML Form Adapter and HTTP Basic Adapter to replace the LDAP Authentication adapter
- Support for the OAuth SAML 2.0 Bearer Assertion Grant Type
- Added an Admin Console Help system updater
- Added IPv6 support

PingFederate 6.5.2 - November 2011

- Security update since the PingFederate 6.5.1 release

PingFederate 6.5.1 - October 2011

- Security updates since the PingFederate 6.5 release

PingFederate 6.5 - August 2011

Note: The PingFederate 6.5 release includes the features described below as well features that were added in a limited-distribution "Preview" release, described in the next section.

- PingFederate now functions as an OAuth 2.0 athorization server
- Added support for Thales (nCipher) nShield Connect HSM
- Account Linking can use an LDAP directory for a persistent data store in addition to a relational database system
- User-Defined Attribute Namespaces can be specified for Browser SSO protocols (similar to what was added to WS-Trust STS) to allow for better Microsoft interoperability
- Adapter to Adapter mapping now counts as a licensed connection
- LDAP Adapter updated to 2.2 with new default web form login template
- Jetty version upgrade from 6.1.7 to 6.1.26

PingFederate 6.5-Preview - April 2011

- Full STS metadata Claims Provider and Relying Party interoperability with Microsoft WIF, WCF, and ADFS 2.0
- Support multiple token-processor instances of the same token type
- Added SAML HoK subject confirmation in the SAML Token Generator
- Added option for STS SAML token KeyInfo to use a signing certificate reference rather than the full signing certificate
- Session-state modifications to support simultaneous and nested SSO transactions
- IdP adapter session handling for IdP adapters that rely on PingFederate for session management to allow for consecutive requests without prompting for credentials

PingFederate 6.4.1 - February 2011

- Corrected license expiration date calculation

PingFederate 6.4 - December 2010

- Support standard .NET WS-Trust Federation Bindings
- Support SAML 2.0 token Holder of Key (HoK) subject confirmation
- Added Metadata Exchange (MEX) endpoint for WIF client to generate bindings automatically for Username, X.509, and SAML tokens
- Added support for WS-Trust 1.4 ActAs property
- Added two-factor authentication capability with the VeriSign® Identity Protection (VIP) Authentication Service Adapter
- OpenToken Adapter 2.4.1 updated to correct issue with Cookie Transport Method and Replay Prevention
- Expanded digital signature secure hash algorithm types - SHA1, SHA256, SHA 384, and SHA512
- The provisioning log can be written to a database—Oracle, Microsoft SQL Server, and MySQL databases supported
- Added SAML protocol support for AuthnContextDeclRefs

PingFederate 6.3 - August 2010

Note: The PingFederate 6.3 release includes the features described below as well features that were added in a limited-distribution "Preview" release, described in the next section.

- PingFederate STS claims-based identity capabilities extended to support interoperability with Microsoft WIF and WCF client frameworks
- Expanded SNMP monitoring variables available in the management information base (MIB)
- Increase the default PingFederate HTTP header buffer size to 8k
- Key stores and key store passwords are dynamically generated per installation
- The default SSL server certificate is generated upon initial startup if an SSL certificate does not exist
- LDAPS trust configuration no longer requires a server restart to take effect

PingFederate 6.3-Preview - April 2010

- Added support for logging to the ArcSight Common Event Format (CEF)
- Added ability to log to a database with failover to file—Oracle, SQL Server, and MySQL databases supported
- Added ability to disable automatic multi-connection validation if the validation time is causing excessive delay
- Extended JDBC Express Provisioning to support MS SQL Server stored procedures
- Added replay prevention capability to the OpenToken IdP Adapter bundled with PingFederate

PingFederate 6.2 - February 2010

- Added IdP-to-SP adapter mapping, which allows user attributes from an IdP adapter to be directly mapped to an SP adapter on the same PingFederate server to create an authenticated session or security context, without the need to generate SAML messages in between
- Provides enhanced logging capabilities including a new audit log, logfilter utility, and ability to log to any accessible file-server directory
- Provides enhanced support for configuration automation including certificate and key management, configuration archive management, and ancillary deployment files
- Extended JDBC Express Provisioning to support MS SQL Server Identity column types
- Added a Logout Endpoint to the LDAP Authentication Adapter
- In clustered mode, the default Inter-Request State Management methodology is now group RPC-based instead of cookie-based
- Added ability to extract CN from DN and extract username from email address for provisioner attributes
- In Luna HSM mode, added the ability to specify the location to store Trusted CA certificates, either in the Sun Java key store or the Luna HSM (see the configuration file `org.sourceid.config.CoreConfig.xml` in the `pingfederate/data/config-store` directory)

PingFederate 6.1 - September 2009

Note: The PingFederate 6.1 release includes the features described below as well features that were added in a limited-distribution "Preview" release, described in the next section.

- Provides support for simplified PingFederate Express connection configuration and export
- Extends support for configuration automation, including listing, copying, and updating features for SaaS Provisioning channels and for Token Translators
- Provides enhanced support for SaaS Provisioning Health and Status Monitoring via JMX
- Provides licensing enhancements including support for organizational licenses, licenses that contain international characters, and web based license import
- Enhances the trust model to include support for anchored certificates, which allows certificates to be included in federation-transaction messaging and used for signature verification if, the given certificate matches the registered Subject DN and is issued by a certificate authority registered as a Trusted CA with PingFederate
- Supports "SP Lite", "IdP Lite", and "e-Gov" Liberty Interoperability profiles for SAML 2.0

PingFederate 6.1-Preview - June 2009

- Provides support for Express Provisioning to a JDBC data source
- Extends support for configuration automation, including listing, copy and update features for data stores and server settings
- Supports certificate-based authentication to the PingFederate administrative console
- Added UI-based data-archive deployment, as well as better error handling for common errors encountered
- Supports mapping of attributes passed in via a WS-Trust STS request to the outgoing token
- Supports logging of a tracking ID (transaction ID) associated with every log entry for a given request to the PingFederate server
- Corrects defects reported by customers in the previous release

PingFederate 6.0 - March 2009

- PingFederate now includes a WS-Trust Security Token Service (STS), enabling organizations to extend identity management to web services. The PingFederate STS shares the core functionality of PingFederate, including console administration, identity and attribute mapping, and certificate security management.
- PingFederate extends support for configuration automation, including connection management and adapter management via the existing command-line tool.
- PingFederate supports enhanced connection based licensing capabilities.
- PingFederate provides transaction based licensing capabilities for evaluation phase license enforcement.
- PingFederate allows administrators to specify the use of a separate certificate that is used for access to the administrative console and a different certificate for runtime processing.
- PingFederate supports configuration of LDAP Groups who are allowed access to the PingFederate Admin application based on PingFederate defined roles.
- PingFederate supports definition of LDAP data stores such that the connection URI for multiple LDAP servers can be specified as the connection string for that LDAP data store.
- PingFederate supports the Virtual List View (VLV) paging mechanism for retrieval of subsets of large result sets returned from the source LDAP data store during the provisioning process. This can significantly enhance performance for retrieval of data from Sun Directory Server (SDS) and similar LDAP servers that support VLV paging.
- PingFederate now stores the PingFederate software version within the configuration store.
- A number of defects reported by customers that existed in the previous release of PingFederate were addressed.

PingFederate 5.3 - December 2008

- PingFederate can be run as a service on Windows 64-bit platforms in addition to Windows 32-bit platforms and Linux platforms.
- PingFederate now supports deployment of SaaS Provisioning plug-ins (JAR files) via a separate installation package.
- PingFederate now supports automating configuration via a command line utility for connection management.
- PingFederate now supports capabilities for monitoring and control of the SaaS Provisioning configuration and data via a command line tool.
- PingFederate now supports validation of certificate revocation information via OCSP.
- PingFederate can now be deployed on Java 6 (JDK 1.6) platforms.
- PingFederate supports access to additional parameters on both the IdP side and the SP side via OGNL expressions.
- PingFederate supports "SP Lite" and "IdP Lite" Liberty Interoperability profiles for SAML 2.0.
- PingFederate supports configuration of the name, domain, and path for the cookie used for conveying state information between servers when cookie-based clustering has been configured.
- A number of past Known Issues and Limitations were addressed.

PingFederate 5.2 - August 2008

- A PingFederate IdP server now provides support for provisioning to selected SaaS providers. PingFederate supports provisioning of user account data from LDAP directories including Active Directory and Sun Directory Server. PingFederate stores synchronization data in JDBC data stores including Hypersonic (for demonstration purposes) and Oracle.
- PingFederate supports quick-connection templates to selected SaaS Providers, including Google Apps, and Salesforce.com

PingFederate 5.1.1 - July 2008

This release corrected several issues, including:

- SP signature verification was failing for assertions containing UTF-8 characters.
- In Windows the PingFederate server was unable to start when the JAVA_HOME system variable contained a space.
- Versions of the OpenToken library were placed in the wrong directory.

- Single Logout (SLO) with two SPs was not being performed for the IdP session(s).
- For SLO with three or more SPs, SP sessions were being stranded.
- Specific to PingFederate 5.1, Custom Data Sources no longer could be used for Adapter Contract fulfillment.
- When testing certain types of OGNL expressions, important error details were being lost when evaluation of these expressions failed.
- For SLO with at least two SPs, under certain circumstances error messages from SPs that did not initiate the SLO were not being processed correctly by the IdP.
- A PingFederate SP instance, when used with the OpenToken Adapter, was converting a plus "+" character to a space " " when constructing the URL for final redirect.
- Signature validation was failing within a PingFederate SP instance when it received an SLO message in which the SAML_SUBJECT was being encrypted.
- PingFederate 5.1 SP instance was no longer supported SiteMinder SSO Zones.

PingFederate 5.1 - April 2008

- The default behavior when PingFederate cannot access a Certificate Revocation List (CRL) is now set correctly. The server no longer treats a non-retrievable CRL as a reason to label certificates as revoked. CRL processing behavior is managed by the revocation-checking-config.xml file in the .
- The IP address to which PingFederate's SNMP agent binds is now controlled by the pf.monitor.bind.address property in the run.properties file.
- Building either of the two example adapters included in the PingFederate SDK no longer fails with an error regarding a missing README.txt.
- The PingFederate server now correctly maintains temporary files within the . The server no longer writes temporary files to the tmp directory of the user running the server.
- Express Provisioning allows user accounts to be created in an LDAP repository and updated directly by an SP PingFederate. User provisioning occurs as part of SSO processing and may be used with any IdP partner.
- The Signature Policy screen in SAML IdP connections contains improved language clarifying how signatures are used to guarantee authenticity of SAML messages.
- The PingFederate package now contains v6.1.7 of Jetty. Jetty is the servlet container used by PingFederate.
- The PingFederate SDK contains a ConfigurationListener interface that may be utilized by developers building adapters. This interface contains methods invoked by the server in response to certain adapter-instance lifecycle events such as creation and deletion.
- Adapter-instance Summary screens now display adapter-instance configuration values specified within a TableDescriptor.
- After the third consecutive failed login attempt, an administrator is blocked from accessing the administrative console for a configurable amount of time (default = 60 seconds).
- When changing an administrator password, the server now forces the new password to differ from the existing password.
- Access to services exposed by the PingFederate server now requires client authentication. These services include Attribute Query, JMX, and Connection Management. An administrator may choose to require client authentication for access to the SSO Directory Service. An ID and Shared Secret comprise the credentials needed for authentication.
- For security, the use of "Expression" in contract fulfillment screens is now disallowed by default. For backward compatibility, customers deploying a configuration archive from a previous version of PingFederate in which expressions were used will continue to have access to expressions. Allowing expressions creates a potential security concern in the PingFederate administrative console.
- The Quick-Start SP Application no longer uses an OGNL expression in fulfilling the SP adapter contract.
- HTTP TRACE requests sent to PingFederate now result in an HTTP 403 Forbidden response.
- By default, "weak" ciphers are no longer supported during SSL handshaking. (For more information as to which cipher suites the server supports, examine the com.pingidentity.crypto.SunJCEManager.xml file.
- It is no longer possible for an administrator to circumvent role and access permissions within the administrative console by direct URL access. The server evaluates HTTP requests for a URL against an administrator's assigned role(s) and responds appropriately.

- The PingFederate runtime server's HTTP listener is now turned off by default. Only messages sent over HTTPS are accepted. This may be controlled in the run.properties file.
- Use of class and package names specific to a PingFederate version were removed from sample source code contained in the PingFederate SDK.
- The /idp/startSSO.ping endpoint now supports an optional ACSIdx query parameter for SAML v2 partners. When provided, the PingFederate IdP attempts to send the SAML Assertion to the Assertion Consumer Service corresponding to the specified Index.
- The initial and maximum JVM heap sizes are set to 256 MB and 1024 MB, respectively, by default. These changes should improve runtime performance on servers with sufficient memory. These settings reside in the run.bat and run.sh files of the .
- During server startup, PingFederate now reports relevant environment variables and adapter- instance information to the server.log.
- Existing partner connections can be deleted through a SOAP call from an external client application.
- The pf-legacy-runtime.war file is no longer deployed by default. This WAR file allows a PingFederate server to continue support of legacy endpoints (those endpoints supported by PingFederate 2). When replacing an existing PingFederate 2 server deployment, manually move this WAR to the .
- The PingFederate server can be configured to support the use of a proxy server when retrieving a CRL from a Certificate Authority. Relevant configuration settings reside in .
- When the PingFederate server relies upon an external LDAP directory to authenticate administrative users, the ldap.password property in the `<pf_install>/pingfederate/bin/ldap.properties` file now supports encrypted credentials.
- The PingFederate server allows imported SSL server certificates containing signatures from one or more intermediate Certificate Authorities. When SSL clients request an SSL connection to the PingFederate server, the entire SSL server certificate chain is presented.
- The Summary and Activation screens for both IdP and SP connections display a valid URL that serves as an example of a startSSO.ping endpoint used by local applications integrating with PingFederate.
- The Web SSO entry screens for both IdP and SP connections include summary information in a table describing relevant configuration settings.
- The PingFederate server prevents auditors from accessing links on the Main Menu that impact external resources. This includes exporting SAML metadata, signing XML files, and creating configuration archives.

PingFederate 5.0.2 - March 2008

- IdP Persistent Reference Cookie (IPRC) — Provides a mechanism allowing an SP PingFederate server to discover a user's IdP based on a persistent browser cookie that contains a reference to the IdP partner previously used for SSO. This feature provides an alternative to standard IdP Discovery for SP-initiated SSO, as defined in the SAML specifications, which uses a common-domain cookie (CDC) written by the IdP (see the PingFederate *Administrator's Manual*). Unlike the IdP Discovery cookie, the IPRC is written by the SP PingFederate each time an SSO event for the user occurs (either IdP- or SP-initiated). The cookie identifies the IdP partner using information in the SAML assertion. For subsequent SP-initiated SSO requests, the SP server can skip a previously required step prompting the user to select an IdP for authentication when multiple IdP partners are configured but none is specifically identified in the SSO call received by the SP PingFederate server.
- Updated the IdP-selection template to make it easier to use. The new selection template is used when no IPRC (or CDC) is available and when there are multiple IdPs to which the user might have previously authenticated.
- Corrected an issue in which a Concurrent Modification Exception was encountered when server clustering is used and debug is turned on for log files. (The workaround for this issue in previous releases is to turn debug off.)
- When no certificate revocation list (CRL) is found during certificate validation checking, the subject certificate is assumed to be valid for the current SSO/SLO transaction. Previously, when no CRL was found, the certificate was deemed invalid and the transaction aborted. The default setting is changed for this release to prevent problems with upgrading to PingFederate 5.x from previous versions.

PingFederate 5.0.1 - January 2008

- Support for rapid provisioning of partner connections is available using Auto-Connect™ technology. Leveraging the existing SAML 2.0 specification, Auto-Connect allows PingFederate deployments to scale easily with minimal

manual involvement. The majority of partner connection configuration occurs at runtime through the exchange of dynamically generated metadata.
- Administrators can authenticate to the administrative console using credentials in an external LDAP directory. This allows organizations with existing admin accounts to provide access to the console without creating and managing individual accounts within PingFederate.
- Partner connections may be created by importing them programmatically into PingFederate through a SOAP interface. This allows administrators to provision partner connections without accessing the administrative console manually. The Connection Management screens (both IdP and SP) contain an "Export" action that creates an XML file containing a connection's configuration.
- Server configuration data may be replicated to a cluster through a SOAP call to the administrative console. This allows cluster deployments to receive configuration changes without accessing the administrative console manually.
- The SAML 2 Attribute Query Profile is supported by PingFederate. This allows SPs to request user attributes from an IdP independent of user authentication.
- Multivalued attributes passed in an Assertion to a WS-Federation partner conform to how ADFS expects them.
- SAML metadata may be generated with a digital signature to guarantee authenticity.
- The Protocol Endpoints popup contains online help links. These links may be used to learn more about the server's endpoints from a partner perspective.
- The User-Session Creation screen in the IdP connection flow contains summary information that provides administrators with insight into the current configuration. Similarly, the Assertion Creation screen in the SP connection flow also provides administrators with useful configuration information.
- Administrators can specify a descriptive "Connection Name" for partner connections.
- The "Web SSO" portion of partner configuration is distinct from first/last-mile configuration.
- Connection summary screens contain +/- buttons to show/hide major sections.
- Metadata import extracts the Base URL from the metadata file and populates relative URLs within the connection.
- PingFederate communicates with an SNMP network-management console using standard SNMP Get and Trap operations.
- The PingFederate engine exposes a URL designed for load balancers to determine whether a PingFederate server is available to process transactions.
- Certificate-management usability is improved.
- Certificate expirations are tracked by PingFederate, and impending expirations may result in a notification sent via email, when configured.
- New, more user-friendly sample applications focus on demonstrating PingFederate server functionality (Quick-Start Guide).
- The entry screen into the "Credentials" area of connections contains useful information about the credentials used.
- The server ID is no longer displayed to the administrator.
- A cluster's administrative console no longer aggregates transactions counts.
- The "Cluster Management" link replaces "High Availability" on the Main Menu.
- Clustering supports TCP and UDP, node authentication, optional encryption, and use of a single port for all communication.
- CRL processing updated to support the U.S. GSA's E-Authentication v2 specification.
- The "About" pop-up contains additional license-key information.

PingFederate 4.4.2 - October 2007

- Mitigated a number of potential security vulnerabilities regarding XML document processing

PingFederate 4.4.1 - June 2007

- Addresses user-interface defects related to attribute query, XML encryption, and LDAP data store lookups

PingFederate 4.4 - May 2007

- Removal of support for the U.S. GSA's E-Authentication v1.0 specification
- Addition for support of signed metadata files

- Support for partner certificate revocation through CRLs
- Increased flexibility around encryption of Name ID in SAML v2.0 SLO requests when the Name ID is encrypted within an assertion
- Support for the SOAP binding for both inbound and outbound SAML v2.0 messages
- Improved support for deployments where the server contains multiple network interfaces
- Usability enhancements to the Main Menu layout and Local Settings flow
- More sophisticated attribute-fulfillment operations through support of a Java-like syntax for data manipulation
- Removal of extraneous credentials settings for WS-Federation and SAML v1.x connections
- Improved display of long connection IDs on the Main Menu
- Inclusion of a demo application that complements the existing sample applications as described in the Quick-Start Guide

PingFederate 4.3 - March 2007

- Virtual Server Identities allow PingFederate to use distinct protocol identifiers in the context of a particular partner connection
- Additional customizable end-user error pages for 'page expired' and general unexpected error conditions
- Increased flexibility by allowing for a list of additional valid hostnames to be used for incoming protocol message validation
- Optionally, the SSO Directory Web Services can be protected with HTTP basic authentication
- New administrative console error page
- Improved short-term state management memory utilization for improved system resiliency
- Improved input-data validation and character-entity encoding of data when displayed--for protection against cross-site scripting attacks
- An IdP connection configured to use only a single SP Adapter Instance will ignore the URL-to-Adapter mapping step at runtime and just use the given adapter
- Blocked directory indexing to limit browsing of static web content
- Disabled unnecessary JRMP JMX port usage
- Mitigated HTTP response splitting attacks by disallowing potentially dangerous characters in all redirects

PingFederate 4.2 - December 2006

- Enhanced transaction logging functionality
- Sensitive user attribute values can be masked in log files to enhance privacy considerations
- The administrative console runs on a distinct port from the runtime engine allowing for more flexible and secure deployment options
- New filtering functionality on connection management screens enables easier management of large numbers of federation partners
- Adapter SDK enhancements to facilitate file downloads
- Usability refinements on X.509 certificate summary screens
- Less verbose description of certificates in drop down boxes improve look and feel
- Multiple partner endpoints of the same type can be configured to use the same binding
- Improved support for reverse proxy deployments

PingFederate 4.1 - October 2006

- Liberty Alliance interoperability certified
- SAML2 x509 Attribute Sharing Profile (XASP)
- Optional Hardware Security Module (HSM) mode, that enables storage of private keys and crypto processing on an external HSM unit that is FIPS-140-2 certified
- Updated Protocol Configuration wizard
- Error handling templates that can be used to build SSO/SLO landing pages that communicate error status and support instructions toned users

- Configuration options that enable multiple, simultaneous authentication profiles for the SOAP back-channel, including HTTP Basic, SSL Client Certificates, and Digital Signatures
- Digital signature capability for client authentication when using SAML 1.x
- Pop-up server endpoint display that filters by role and configurations made
- Two digital signature verification certificates can be assigned to a connection, allowing the partner flexibility in selecting one certificate or the other. When one certificate expires, the other certificate is used without the need for close synchronization.
- A `run.properties` configuration that allows an admin to specify an alternate port with which to communicate over the back-channel to partner's SAML gateway
- Support for 32-and 64- bit machine architectures

PingFederate 4.0 - June 2006

- Deploy multiple adapters as an IdP to look up different session security contexts across security domains and applications
- Save a partially completed connection as a draft
- Copy a connection to rapidly set up other partners or test environments with similar configurations
- Attribute source SDK enables retrieval of attributes from additional data source interfaces such as SOAP, flat files, or custom interfaces
- Multi-administrator support
- Ability to edit SP adapters that are in-use with target systems
- Encrypt or decrypt entire assertions or select elements, of particular value when intermediaries may handle SAML traffic
- Generate unique, Transient Name Ids each time the user federates to protect their identity
- SAML 2-compliant IdP Discovery mechanism that enables an SP to dynamically determine the appropriate IdP for the user
- Integration Kits provide additional methods that streamline passing of authentication context from an IdP to an SP
- Single log-out across all connections and protocols that support SLO
- Using an affiliate id, an SP can instruct an IdP to re-use the same persistent name identifier that was already used at other applications within the portal
- Non-normative support for SP-initiated SSO with SAML 1.x protocols

PingFederate 3.0.2 - February 2006

- Upgrade of Jetty component to v5.1.10 in response to a security warning from the National Vulnerability Database

PingFederate 3.0.1 - December 2005

- Complete clustering support
- Optional email notification on licensing issues
- You can edit a previously configured connection (either IdP or SP) that uses a data store that is unavailable

PingFederate 3.0 - November 2005

- Support for SAML 2.0
- Use-case wizard for partner connection configurations
- Support for multiple security domains
- Redesigned user interface
- Embedded clustering
- Fixes for LDAP and JDBC connectivity

PingFederate 2.1 - July 2005

- Patched a concurrency bug in the XML security library
- Patched a memory leak in the XML-to-object binding library

- Removed the core protocol processor's reliance on a workflow engine to resolve a memory leak and improve overall performance
- Fixed a subtle memory leak in the module that tracks assertions in order to prevent replay in the POST profile
- Updated the default server SSL certificate (extended the expiration date)

PingFederate 2.0 - February 2005

- Initial release

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 9.1.4.

Both the platform-specific installer 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, both tools do not affect the source installation.

Note: If you are upgrading from PingFederate 5.3 (or an earlier version), contact *support@pingidentity.com* 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 UNIX/ Linux	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 installer 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, both tools do not affect 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.
- If you have not already done so, obtain a new license key as needed.
- If you have not recently updated the Oracle Java SE Runtime Environment (Server JRE) to version 8 on your PingFederate servers, you must do so.

  When you upgrade the Server JRE, 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 Java 8. 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

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 9.1.4 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@pingidentity.com* for more information.

Both the platform-specific installer 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, both tools do not affect 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.

- 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*.

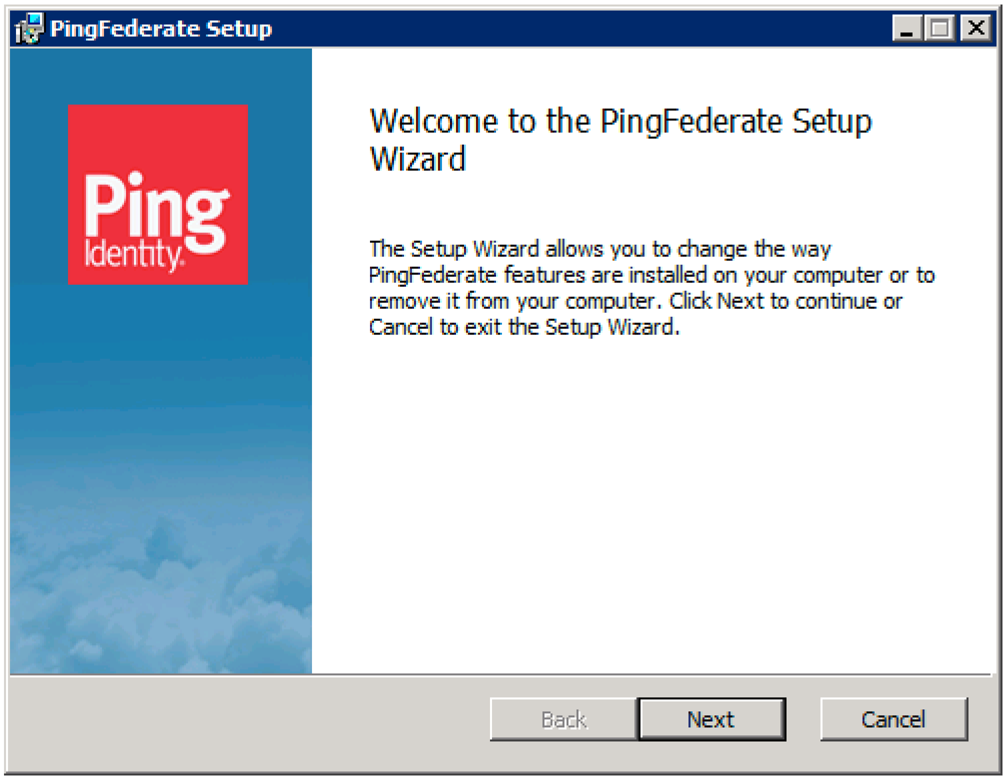
Upgrade PingFederate using the installer

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.

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.



Welcome to the PingFederate Setup Wizard

The Setup Wizard allows you to change the way PingFederate features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Errors are rare but 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 **Server Configuration** > **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen.

Upgrade PingFederate using the Upgrade Utility on Windows

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.

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 **Server Configuration** > **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **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 30).
   b) Install the new PingFederate service (see *Install the PingFederate service on Windows manually* on page 27).

   For a clustered PingFederate environment, re-install the PingFederate service on all nodes.

Upgrade PingFederate on Red Hat Enterprise Linux

On a Red Hat Enterprise Linux server, you may upgrade your PingFederate installation from version 6.0 (or a more recent version) to version 9.1.4 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 Windows 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@pingidentity.com* for more information.

Both the platform-specific installer 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, both tools do not affect 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.

- 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

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.

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 **Server Configuration** > **Cluster Management** screen.
7. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen.

Upgrade PingFederate using the Upgrade Utility on Red Hat Enterprise Linux

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.

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 **Server Configuration** > **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **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 the PingFederate service on Linux manually* on page 27).

   PingFederate SysV initialization script

   Edit the PingFederate SysV initialization script and reconfigure the PingFederate service (see *step 8* in *Install the PingFederate service on Linux manually* on page 27).

Upgrade PingFederate on UNIX/Linux

On a UNIX or Linux server, you may upgrade your PingFederate installation from version 6.0 (or a more recent version) to version 9.1.4 by using the PingFederate Upgrade Utility. (They are both command-line tools.)

Note: If you are upgrading from PingFederate 5.3 (or an earlier version), contact *support@pingidentity.com* 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.

- 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 UNIX/Linux

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.

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 **Server Configuration** > **Cluster Management** screen.
8. If you are upgrading a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **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 the PingFederate service on Linux manually* on page 27).

   PingFederate SysV initialization script

   Edit the PingFederate SysV initialization script and reconfigure the PingFederate service (see *step 8* in *Install the PingFederate service on Linux manually* on page 27).

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.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

Review the following post-upgrade tasks:

- *Copy customized files or settings* on page 756
- *Review database changes* on page 758
- *Review log configuration* on page 761
- *Migrate other components* on page 762
- *Reset files and variable for HSM* on page 764

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 can simply copy over the relevant files from the Jetty configuration directory, `<pf_install>/pingfederate/etc`.

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 265.

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 776 and *memoryoptions and upgrade* on page 773

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) to version 9.1.4, 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 9.1.4, 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 or UNIX. For more information about the `provmgr` command-line tool, see *Outbound provisioning CLI* on page 130.

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 **Server Configuration** > **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 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 4.0 (or older) drivers, 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 **Server Configuration** > **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 4.1 drivers. 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

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 following resources:

- *Upgrading from PingFederate 8.x or 9.0.x* on page 762
- *Upgrading from PingFederate 6.x or 7.x* on page 762

Upgrading from PingFederate 8.x or 9.0.x

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:

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

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:

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, see:

- *Update custom authentication selector* on page 762
- *Migrate to the integrated LDAP Username PCV* on page 763
- *Migrate to the integrated Username Token Processor* on page 763

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

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.

1. Create an instance of the integrated LDAP Username PCV.
   a) On the **Server Configuration** > **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 **Server Configuration** > **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

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.

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 Thales 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 9.1.4 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

# Enables fast recycling of TIME_WAIT sockets.
# (While this may increase performance, it can introduce issues with
# persistent keep-alive connections when interacting with upstream/
# downstream network appliances (load balancers, proxies, etc.).
# This setting should only be enabled after the system administrator
# reviews security considerations.)
net.ipv4.tcp_tw_recycle=0

# 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=0

# 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

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.

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

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 769). 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

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

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.

1. For JDBC data stores:
   a) On the **Server Configuration** > **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 768).
2. For LDAP data stores:
   a) On the **Server Configuration** > **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.

   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 **Server Configuration** > **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 771).

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 776 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	• The installer or the install script creates a new PingFederate installation.

Installation medium	Expected behavior
or PingFederate install script for Red Hat Enterprise Linux	• 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 or UNIX** • 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*	• 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.

Upgrade path	Expected behavior when the `jvm-memory.options` file does not exist in the source installation
PingFederate install script for Red Hat Enterprise Linux	• 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.
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.
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.

Upgrade path	Expected behavior when the `jvm-memory.options` file exists in 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

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.

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 56.)

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)

Index