PingFederate®

Version 8.1.4

Copyright

PingFederate manuals

Version 8.1.4
July 21, 2016

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

Trademark

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 online documentation at *documentation.pingidentity.com* 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 in this date: **July 21, 2016**.

Contents

Get started with PingFederate..... 7

Introduction to PingFederate..... 7
- About identity federation and SSO..... 7
- Security token service..... 8
- OAuth authorization server..... 9
- User account management..... 9
- Enterprise deployment architecture..... 10
- Additional features..... 10

Installation..... 11
- Deployment options..... 12
- System requirements..... 13
- Port requirements..... 16
- Install Java..... 17
- Installation options..... 17
- Open the PingFederate administrative console..... 20
- Initial setup wizard..... 20
- Install PingFederate as a service..... 23
- Uninstall PingFederate..... 25

The administrative console..... 26
- Tasks and steps..... 27
- Console buttons..... 27

Supported standards..... 28
- Federation roles..... 28
- Terminology..... 29
- Browser-based SSO..... 30
- Web Services standards..... 45
- OAuth 2.0..... 47
- System for Cross-domain Identity Management (SCIM)..... 50
- Transport and message security..... 50

Supported hardware security modules..... 51
- Install and configure Luna SA client and PingFederate..... 51
- Install and configure nShield Connect client and PingFederate..... 53

Administrator's manual..... 56

Key concepts..... 56
- Connection types..... 57
- About WS-Trust STS..... 57
- PingFederate OAuth AS..... 60
- SSO integration kits and adapters..... 64
- Hierarchical plug-in configurations..... 66
- Identity mapping..... 66
- About attributes..... 67
- Security infrastructure..... 72
- Using Auto-Connect..... 75
- User provisioning..... 78
- Federation planning checklist..... 79
- Federation hub..... 83

System administration..... 87

Manage PingFederate license ... 88
PingFederate log files ... 89
Export metadata to an XML file ... 101
Sign XML files ... 103
Replicate configuration ... 103
Manage configuration archives ... 103
Account management ... 105
Alternative console authentication ... 108
Manage email configuration ... 110
Virtual host names ... 111
PingFederate properties ... 112
Automating configuration migration ... 114
Outbound provisioning CLI ... 117
Customizable user-facing screens ... 120
Configure password policy ... 125
Manage expired persistent grants ... 126
Extend the lifetime of the PF cookie ... 126
Configure forward proxy server settings ... 127
Add custom HTTP response headers ... 127
Customize the heartbeat message ... 128
Customize the favicon for application and protocol endpoints ... 128
Configure the behavior of searching multiple data stores with one mapping ... 128
System settings ... 129
Server settings ... 129
Connect to PingOne from Server Configuration ... 142
Manage data stores ... 143
IdP discovery ... 155
Configure redirect validation ... 158
Security management ... 160
Certificate management ... 160
Authentication ... 172
OAuth configuration ... 181
Enable the OAuth AS ... 181
Configure OAuth settings ... 182
Configure an OAuth SAML Grant IdP connection ... 204
OAuth attribute mapping using a data store ... 207
Authentication policies ... 208
Selectors ... 208
Policies ... 215
Policy contracts ... 235
Adapter Mappings ... 236
Identity provider SSO configuration ... 238
IdP application integration settings ... 238
View IdP protocol endpoints ... 244
Manage SP connections ... 245
Define SP affiliations ... 305
Configure SP Auto-Connect ... 306
Service provider SSO configuration ... 308
SP application integration settings ... 309
Federation settings ... 316
Manage IdP connections ... 318
Configure IdP Auto-Connect ... 376
WS-Trust STS configuration ... 378
Server settings ... 378
IdP configuration for STS ... 381
SP configuration for STS ... 402

IdP-to-SP bridging ... 417
Adapter-to-adapter mappings ... 417
Token translator mappings ... 421
Bundled adapters ... 424
Kerberos Adapter ... 424
HTML Form Adapter ... 428
HTTP Basic Adapter ... 432
OpenToken Adapter ... 434
Composite Adapter ... 439
Application endpoints ... 442
IdP endpoints ... 442
SP endpoints ... 444
System-services endpoints ... 452
OAuth 2.0 endpoints ... 455
Token endpoint ... 455
Token revocation endpoint ... 459
Authorization endpoint ... 460
Grant-management endpoint ... 464
OpenID Connect provider metadata endpoint ... 464
Web Service interfaces ... 465
Connection Management Service ... 466
SSO Directory Service ... 468
SOAP request and response examples ... 470
OAuth Client Management Service ... 471
OAuth Access Grant Management Service ... 477
Session revocation API endpoint ... 478
PingFederate administrative API ... 480
Attribute mapping expressions ... 485
Enable and disable expressions ... 485
Construct OGNL expressions ... 486
Using the OGNL edit screen ... 488
Customize assertions and authentication requests ... 489
Message types and available variables ... 489
Sample customizations ... 492
Fulfillment by data store queries ... 494
Attribute mapping with multiple data sources ... 494
Data store query configuration ... 495
Troubleshooting ... 500
Server startup ... 501
Data store issues ... 501
Troubleshooting SSO or SLO failures ... 501
Other runtime issues ... 505
Glossary ... 505
List of acronyms ... 513

Server clustering guide ... 515
Overview of clustering ... 515
Deploy cluster servers ... 516
Configure clustering properties ... 516
Deployment architecture ... 519
Runtime state-management services ... 523
Deploy provisioning failover ... 528
Configuration synchronization ... 529
Console configuration push ... 529
Configuration-archive deployment ... 530

SSO integration overview..... 531

Integration introduction..... 531
SSO integration concepts..... 531
Identity provider integration..... 532
Service provider integration..... 534
Summary..... 536

SDK developer's guide..... 539

Preface..... 539
SDK introduction..... 540
Get started with the SDK..... 541
- Directory structure..... 541
- Set up your project..... 541

Implementation guidelines..... 542
- Shared interfaces..... 542
- Implement an IdP adapter..... 543
- Implement an SP adapter..... 545
- Implement a token processor..... 547
- Implement a token generator..... 548
- Implement an authentication selector..... 548
- Implement a custom data source..... 549
- Implement a password credential validator..... 550
- Implement an identity store provisioner..... 551
- Build and deploy your project..... 556

Release notes..... 559

PingFederate 8.1.4 — July 2016..... 559
Upgrade considerations..... 559
Deprecated features..... 563
Known issues and limitations..... 563
Previous releases..... 566
- PingFederate 8.1.3 — May 2016..... 566
- PingFederate 8.1.2 — April 2016..... 566
- PingFederate 8.1.1 — February 2016..... 567
- PingFederate 8.1 — January 2016..... 568
- PingFederate 8.0.4 — November 2015..... 571
- PingFederate 8.0.3 — September 2015..... 571
- PingFederate 8.0.2 — August 2015..... 572
- PingFederate 8.0.1 — July 2015..... 573
- PingFederate 8.0 — June 2015..... 573
- PingFederate 7.3 — January 2015..... 575
- PingFederate 7.2 R2 — September 2014..... 575
- PingFederate 7.2.1 — August 2014..... 576
- PingFederate 7.2 — June 2014..... 576
- PingFederate 7.1.4 — June 2014..... 576
- PingFederate 7.1 R3 — March 2014..... 577
- PingFederate 7.1.3 — February 2014..... 577
- PingFederate 7.1.2 — December 2013..... 577
- PingFederate 7.1 R2 — December 2013..... 577
- PingFederate 7.1.1 — November 2013..... 578
- PingFederate 7.1 — August 2013..... 578
- PingFederate 7.0.1 — May 2013..... 578

PingFederate 7.0 — April 2013 ... 578
PingFederate 6.11 — December 2012 ... 579
PingFederate 6.10.1 — January 2013 ... 579
PingFederate 6.10 — September 2012 ... 579
PingFederate 6.9 — June 2012 ... 580
PingFederate 6.8 — April 2012 ... 580
PingFederate 6.7 — February 2012 ... 580
PingFederate 6.6 — December 2011 ... 580
PingFederate 6.5.2 — November 2011 ... 580
PingFederate 6.5.1 — October 2011 ... 580
PingFederate 6.5 — August 2011 ... 580
PingFederate 6.5-Preview — April 2011 ... 581
PingFederate 6.4.1 — February 2011 ... 581
PingFederate 6.4 — December 2010 ... 581
PingFederate 6.3 — August 2010 ... 581
PingFederate 6.3-Preview — April 2010 ... 581
PingFederate 6.2 — February 2010 ... 581
PingFederate 6.1 — September 2009 ... 582
PingFederate 6.1-Preview — June 2009 ... 582
PingFederate 6.0 — March 2009 ... 582
PingFederate 5.3 — December 2008 ... 583
PingFederate 5.2 — August 2008 ... 583
PingFederate 5.1.1 — July 2008 ... 583
PingFederate 5.1 — April 2008 ... 584
PingFederate 5.0.2 — March 2008 ... 585
PingFederate 5.0.1 — January 2008 ... 585
PingFederate 4.4.2 — October 2007 ... 586
PingFederate 4.4.1 — June 2007 ... 586
PingFederate 4.4 — May 2007 ... 586
PingFederate 4.3 — March 2007 ... 587
PingFederate 4.2 — December 2006 ... 587
PingFederate 4.1 — October 2006 ... 587
PingFederate 4.0 — June 2006 ... 588
PingFederate 3.0.2 — February 2006 ... 588
PingFederate 3.0.1 — December 2005 ... 588
PingFederate 3.0 — November 2005 ... 588
PingFederate 2.1 — July 2005 ... 588
PingFederate 2.0 - February 2005 ... 588

PingFederate Upgrade Utility user guide ... 589

Upgrading PingFederate ... 589
Prepare for the upgrade ... 590
Run the PingFederate Upgrade Utility ... 590
Review post-upgrade tasks ... 593
Known limitations ... 597

Download documentation ... 599

Legal information ... 600

Get started with PingFederate

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 7 — A high-level view of federated identity, secure web SSO, and PingFederate features.
- *Installation* on page 11 — How to install PingFederate and run the administrative console for the first time.
- *The administrative console* on page 26 — A primer on using the administrative console and configuration screens.
- *Supported standards* on page 28 — An overview of industry standards that PingFederate supports, including the Security Assertion Markup Language (SAML) and WS-Federation.
- *SafeNet Luna SA HSM* — How to install and configure PingFederate with the SafeNet Luna SA HSM as part of compliance with the Federal Information Processing Standard (FIPS) 140-2.
- *Thales nShield Connect HSM* — How to install and configure PingFederate with the Thales nShield Connect 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, 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, and OAuth).

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 28.

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.

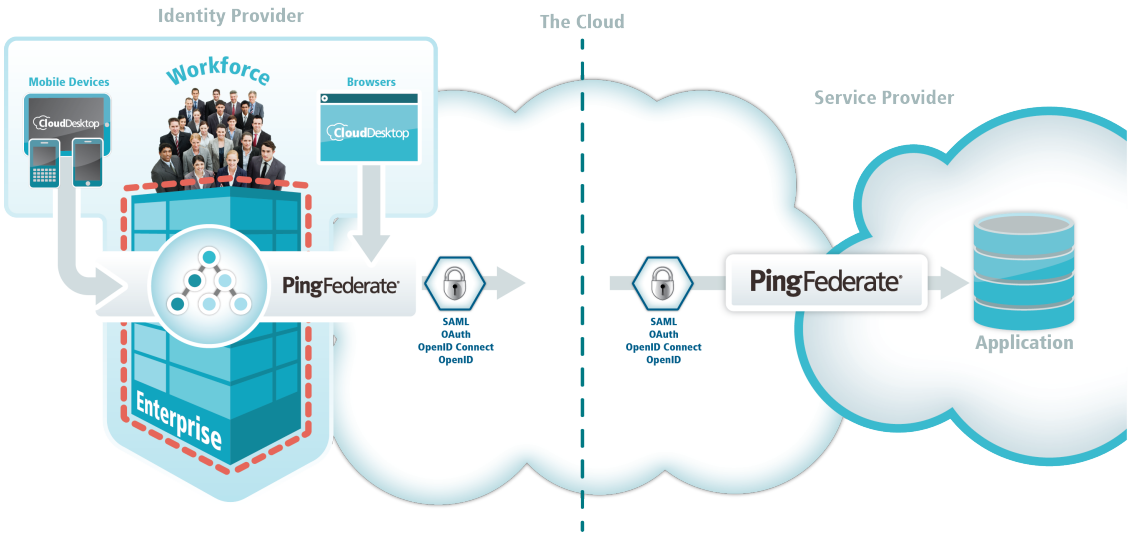


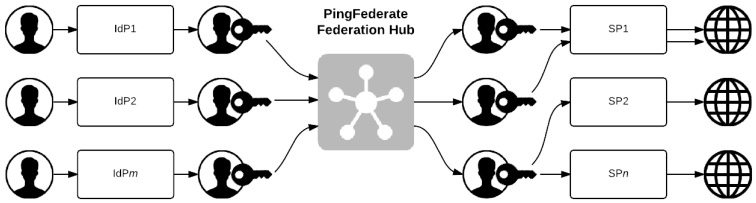
Figure 1: Secure Single Sign-on

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

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

Identity 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.



For more information, see *Federation hub* on page 83 in the “Key Concepts” chapter of the PingFederate *Administrator's manual*.

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 45).

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 57 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*).

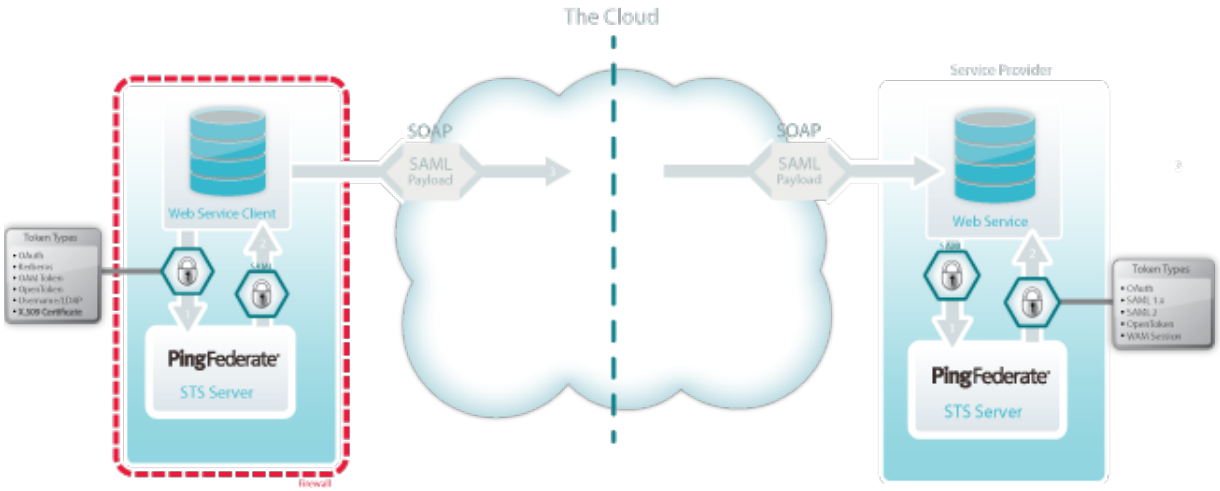


Figure 2: Security Token Service SSO

OAuth authorization server

PingFederate can act as an OAuth Authorization Server (OAuth 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* on page 47).

The PingFederate OAuth AS can be configured independently or in conjunction with STS and browser-based SSO for either an IdP or an SP deployment. For more information, see *PingFederate OAuth AS* on page 60 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.

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 78 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.

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 78 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*).

Additional features

PingFederate's 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** – These tailored kits simplify integration with existing applications while minimizing impacts on existing infrastructure.
- **Token Translators** – These specialized plug-ins connect the STS with Web Service Providers and Clients to enable access to identity-enabled Web Services, which may require a range of different token types.
- **SaaS Connectors** – These plug-ins provide quick-connection templates and automated user provisioning and deprovisioning for selected SaaS providers, including Salesforce and Google Apps.
- **Cloud Identity Connectors** – These plug-ins allow cloud identity providers such as Facebook, Yahoo!, Google and Salesforce to authenticate and connect users to SSO-enabled applications.

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 web site*, take only minutes to install, and are configured from within the PingFederate administrative console.

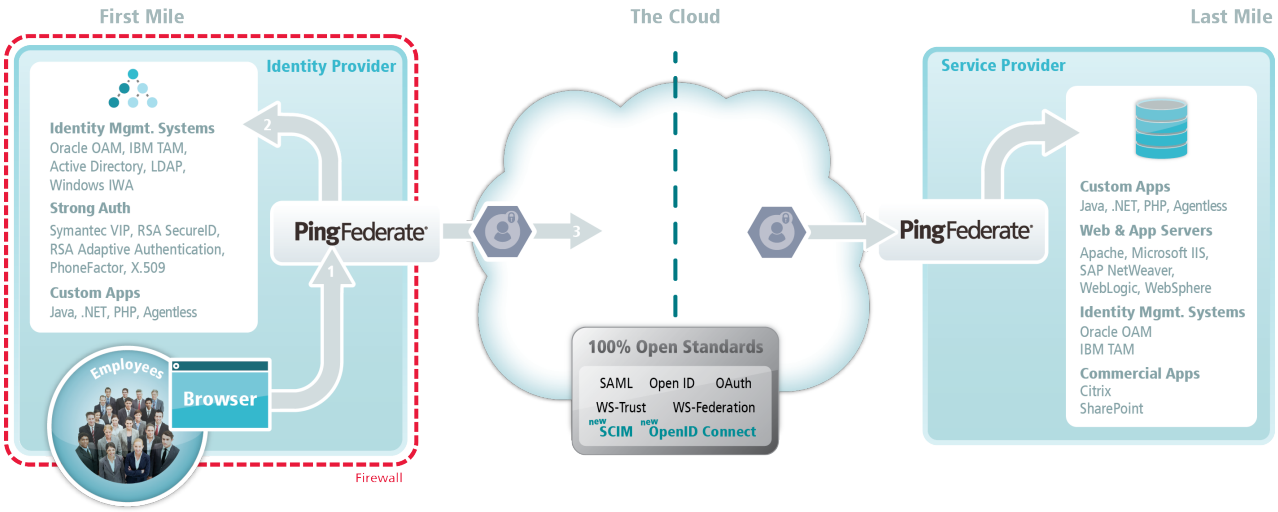


Figure 3: Multiple security-domain, multi-protocol federation

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

For more information, see *SSO integration kits and adapters* on page 64 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.

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 58 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.

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 78 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*). 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, Windows Live, 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 *http://www.pingone.com*.

Installation

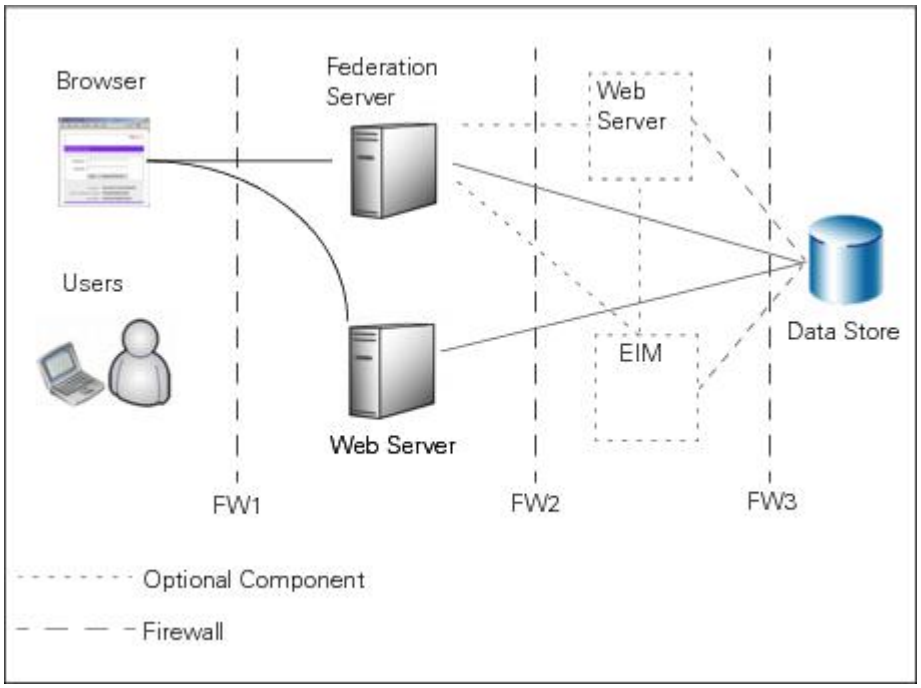
PingFederate® is packaged as a stand-alone server based on J2EE application server technology. A new installation involves the following tasks:

- *Deployment options* on page 12
- *System requirements* on page 13
- *Port requirements* on page 16
- *Install Java* on page 17
- *Installation options* on page 17
- *Initial setup wizard* on page 20

Deployment options

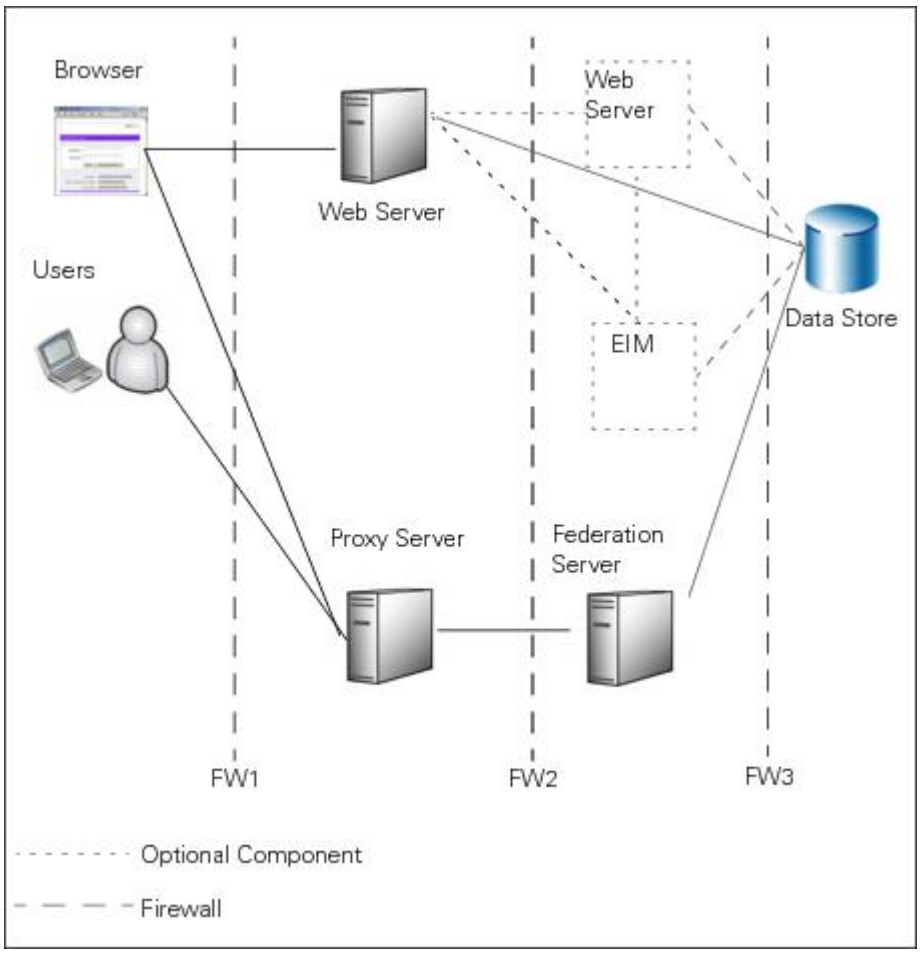
There are many options for deploying PingFederate® in your network environment, depending on your needs and infrastructure capabilities.

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

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.

- Microsoft Windows Server 2008 R2 SP1
- Microsoft Windows Server 2012 Standard
- Microsoft Windows Server 2012 R2 Datacenter
- Oracle Enterprise Linux 6.7 (Red Hat compatible kernel)
- Oracle Enterprise Linux 7.1 (Red Hat compatible kernel)
- Oracle Solaris 10
- Red Hat Enterprise Linux ES 6.7
- Red Hat Enterprise Linux ES 7.1

- SUSE Linux Enterprise 11 SP4

Virtual Systems

Although Ping Identity does not qualify or recommend any specific virtual-machine (VM) products, PingFederate has been shown to run well on several, including VMWare, Xen, and Windows Hyper-V.

Note: This 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 and/or interoperability of any VM software with its products.

Java environment

- Oracle Java SE Runtime Environment (Server JRE) 8 update 72 (64-bit)

Supported browsers for end users

- Chrome
- Edge
- Firefox
- Internet Explorer (version 9 and higher)
- Safari
- Android 6
- iOS 9

Supported browsers for the administrative console

- Chrome
- Firefox
- Internet Explorer (version 11 and higher).

Data store integration

Note: The following lists include data stores tested against specific PingFederate use cases. However, comparable data stores for each feature may potentially include any LDAP v3-compliant directory service or JDBC-compatible database.

For user-attribute lookup

- Microsoft Active Directory (2008 R2 and 2012)
- Oracle Directory Server Enterprise Edition 11g
- UnboundID Data Store 5.1
- Microsoft SQL Server (2012 and 2014)
- Oracle Database (11g R2 and 12c)
- Oracle MySQL 5.6

For outbound provisioning

- Provisioning channel data source
  - Microsoft Active Directory (2008 R2 and 2012)
  - Oracle Directory Server Enterprise Edition 11g
  - UnboundID Data Store 5.1
- Provisioning internal data store
  - Microsoft SQL Server (2012 and 2014)
  - Oracle Database (11g R2 and 12c)
  - Oracle MySQL 5.6

For inbound provisioning

- Microsoft Active Directory (2008 R2 and 2012)

For just-in-time (JIT) provisioning (external target database)

- Microsoft SQL Server (2012 and 2014)

For account linking

- Microsoft Active Directory (2008 R2 and 2012)
- Oracle Directory Server Enterprise Edition 11g
- UnboundID Data Store 5.1
- Microsoft SQL Server (2012 and 2014)
- Oracle Database (11g R2 and 12c)
- Oracle MySQL 5.6

For OAuth client configuration:

- Microsoft SQL Server (2012 and 2014)
- Oracle Database (11g R2 and 12c)
- Oracle MySQL 5.6

For OAuth persistent grants:

- Microsoft SQL Server (2012 and 2014)
- Oracle Database (11g R2 and 12c)
- Oracle MySQL 5.6
- Microsoft Active Directory 2012
- Oracle Directory Server Enterprise Edition 11g
- UnboundID Data Store 5.1
- Custom implementation through the PingFederate SDK

Note: PingFederate has been tested with vendor-specific JDBC 4.1 (or higher) drivers. To obtain your database driver JAR file, contact your database vendor.

Hardware security module (optional)

- SafeNet Luna SA version 5.4 (or higher)

  Client Driver Version: 5.4 (or higher)

- Thales nShield Connect version 2.51.10

  Client Driver Version: 11.70

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
- 300 MB 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
- 750 MB 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	Descripti
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	Administration engine	Used for i administra Configura `run.pro`
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	Used for i runtime en Configura `run.pro`
Cluster traffic (TCP)	JGroups, inbound, TCP, 7600	PingFederate peer servers in a clustered PingFederate environment	PingFederate	Used for c engine no transport r set to TCP Configura `run.pro`
Cluster traffic (TCP)	JGroups, inbound, TCP, 7700	PingFederate peer servers in a clustered PingFederate environment	PingFederate	Used by o as part of detection transport r set to TCP Configura `run.pro`
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 a optional d mechanism

[1] 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	Descripti
Cluster traffic (UDP)	JGroups, inbound,UDP, 7601	PingFederate peer servers in a clustered PingFederate environment	PingFederate	Used for c engine no the transp traffic is s transport r Configura `run.pro`
PingOne integration (optional)	HTTPS and secure WebSocket, TCP, 443	PingFederate Applicable to the console node in a clustered PingFederate environment	pingone.com	Used for c PingFeder the purpos and maint connectio of PingFe authentica PingOne c
Active Directory domains/Kerberos realms (optional)	Kerberos, outbound, TCP or UDP, 88	PingFederate	Windows domain controllers	Used for c PingFeder controller Kerberos

Note: Depending on the integration kits deployed and the connecting third-party systems, 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* 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 download web site* (`www.oracle.com/technetwork/java/javase/downloads/index.html`).

1. Download and install the Server JRE from *www.oracle.com/technetwork/java/javase/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 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 install PingFederate® by extracting the distribution ZIP file or running a platform-specific installer (for Microsoft Windows Server or Red Hat Enterprise Linux).

Important: If your site requires compliance with FIPS 140-2, or plans on managing keys and certificates using a supported hardware security module (HSM), you must install PingFederate by a distribution ZIP file.

Install PingFederate on Windows

1. Request a license key via the *Ping Identity licensing web page* (`www.pingidentity.com/support-and-downloads/licensing.cfm`).
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 17).

   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.
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. **Important:** If your site requires compliance with FIPS 140-2, or plans on managing keys and certificates using a supported hardware security module (HSM), you must install PingFederate by a distribution ZIP file.

   Note: 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, in certain server-clustering scenarios), either install each instance in a different location or rename the parent directory to install a parallel file structure in the same location.

   The installation directory is referred as `<pf_install>`, where the `pingfederate` directory can be found; for example, `<pf_install>/pingfederate`.
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 started in <X>s:<Y>ms
   ```

   Tip: To configure PingFederate to run as a service, follow the steps in *Install the PingFederate service on Windows manually* on page 23.

Install PingFederate on Red Hat Enterprise Linux

1. Request a license key via the *Ping Identity licensing web page* (`www.pingidentity.com/support-and-downloads/licensing.cfm`).
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 17).
4. Install PingFederate via the platform-specific installer or the distribution ZIP file.

Installation media	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. **Important:** If your site requires compliance with FIPS 140-2, or plans on managing keys and certificates using a supported hardware security module (HSM), you must install PingFederate by a distribution ZIP file.

Note: 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, in certain server-clustering scenarios), either install each instance in a different location or rename the parent directory to install a parallel file structure in the same location.

The installation directory is referred as `<pf_install>`, where the `pingfederate` directory can be found; for example, `<pf_install>/pingfederate`.

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 started in <X>s:<Y>ms
   ```

Tip: To configure PingFederate to run as a service, follow the steps in *Install the PingFederate service on Linux manually* on page 24.

Install PingFederate on UNIX/Linux

Refer to *System requirements* on page 13 for a list of qualified UNIX and Linux operating systems.

1. Request a license key via the *Ping Identity licensing web page* (`www.pingidentity.com/support-and-downloads/licensing.cfm`).
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 17).
4. Download and extract the distribution ZIP file into an installation directory.

   Note: 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, in certain server-clustering scenarios), either install each instance in a different location or rename the parent directory to install a parallel file structure in the same location.

   The installation directory is referred as `<pf_install>`, where the `pingfederate` directory can be found; for example, `<pf_install>/pingfederate`.

5. If your site requires compliance with FIPS 140-2, or plans on managing keys and certificates using a supported hardware security module (HSM), see *Supported hardware security modules* for additional installation information.
6. 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 started in <X>s:<Y>ms
```

Tip: To configure PingFederate to run as a service, follow the steps in *Install the PingFederate service on Linux manually* on page 24.

Open the PingFederate administrative console

The administrator's user interface for PingFederate®, the *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 a web browser; for examples, Chrome, Firefox, or Internet Explorer 11.
2. Browse to the following URL:

   ```
   https://<DNS_NAME>:9999/pingfederate/app
   ```

   where `<DNS_NAME>` is the fully qualified name of the machine running the PingFederate server.

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 20 (optional)
- *Review or import your license* on page 21
- *Enter the basic information* on page 21
- *Select your federation roles* on page 21
- *Configure identity provider settings* on page 21 (optional)
- *Create an administrator account* on page 23
- *Review the initial configuration* on page 23

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 to PingOne, a *managed* PingOne SP connection is created for you. In this managed SP connection, PingFederate automatically uploads any metadata changes to PingOne. For example, if you update your **Base URL** under Server Settings or extend the attribute contract to PingOne, those changes will be propagated to PingOne without additional effort. In addition, PingFederate will automatically download new signing certificates from PingOne and update the connection accordingly.

1. Click **Sign on to PingOne to get your activation key**.
2. Sign on using your PingOne admin portal credentials.
3. Copy the **Activation Key** value.
4. Click **Finished**.
5. Close the browser tab and go back to the PingFederate administrative console.
6. In 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

- 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 in the **License** screen.

  Alternatively, you can import your license file in the **Server Configuration** > **License Management** screen *before* the trial license expires.
- If you have opted to setup PingFederate without PingOne, import your license in the **License** screen.

  Note: If you do not have a PingFederare license yet, request a license key via the *Ping Identity licensing web page* (`www.pingidentity.com/support-and-downloads/licensing.cfm`) or contact *sales@pingidentity.com*.

Click **Next** to continue.

Enter the basic information

In 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

In 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 selected 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 selected. 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* 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 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* 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*.

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

- In 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

Use the **Administrator Account** screen to 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

In 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.

In the **Complete** screen, click **Next** to apply the configuration, and then click **Done** to exit to the Main menu.

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 install 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: When you start the server manually, you must also 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, PingFederate has already been configured to run as a service and started automatically at the end of the installation process. That is, you do not need to install the PingFederate service manually.

1. Install PingFederate.

   Note: Ensure `JAVA_HOME` and `PATH` are set as system variables (see *Install Java* on page 17).

2. Ensure you are logged on with full Administrator privileges.
3. Start PowerShell or Command Prompt as an Administrator.
4. In PowerShell or Command Prompt, run the `install-service.bat` file to install the service on one of the following platforms:

   On a 64-bit Windows x86 platform, run `install-service.bat` from the directory: `<pf_install>\pingfederate\sbin\win-x86-64`

   Or:

   On a 64-bit Windows Itanium platform, run `install-service.bat` from the directory:

   `<pf_install>\pingfederate\sbin\win-itanium-64`
5. Open the **Control Panel** > **Administrative Tools** > **Services** management console.
6. 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, PingFederate has already been configured to run as a service and started automatically at the end of the installation process. That is, you do not need to install the PingFederate service manually.

To run PingFederate as a service on Linux, you must place a script in the system initialization directory.

Note: This document uses `/etc/rc.d/init.d/` as the system initialization directory. Modify accordingly if the system initialization directory resides elsewhere on your Linux server.

1. Install PingFederate.
2. Log on as root.
3. Create a new user account for the service.

   For this procedure, the variable `<pf_user>` is used to refer to this account.

   Note: Ensure the environment variable `JAVA_HOME` is set and the `PATH` variable updated for `<pf_user>` (see *Install Java* on page 17).
4. Change the PingFederate installation directory (`<pf_install>`) ownership and modify the read and write permissions:

   ```
   chown -R <pf_user> <pf_install>
   ```

   ```
   chmod -R 775 <pf_install>
   ```
5. Place the code below into a file called `<pf_user>` in the directory:

   `/etc/rc.d/init.d/`

   Note: Replace instances of `<pf_user>` and `<pf_install>` in the script below, and in the commands that follow, with their respective values.

```
#! /bin/sh

   start(){
      echo "starting PingFederate.."
      su - <pf_user> \
      -c '<pf_install>/pingfederate/sbin/pingfederate-run.sh \
      > /dev/null 2> /dev/null'
   }
   stop(){
      echo "stopping PingFederate.."
```

```
      su - <pf_user> \
      -c '<pf_install>/pingfederate/sbin/\
         pingfederate-shutdown.sh'
   }

   restart(){
      stop
   # padding time to stop before restart
      sleep 60
   # To protect against any services that are not stopped,
   # uncomment the following command.
   # (Warning: this kills all Java instances running as
   # <pf_user>.)
   #   su - <pf_user> -c 'killall java'
      start
   }
   case "$1" in
      start)
         start
         ;;
      stop)
         stop
         ;;
      restart)
         restart
         ;;
      *)
      echo "Usage: <pf_user> {start|stop|restart}"
      exit 1
   esac
   exit 0
```

6. Create symbolic links using commands listed below.

   The links specify the order in which PingFederate starts and stops.

   ```
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc3.d/S84<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc5.d/S84<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc4.d/S84<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc6.d/K15<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc0.d/K15<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc1.d/K15<pf_user>
   ln -s /etc/rc.d/init.d/<pf_user> /etc/rc2.d/K15<pf_user>
   ```

7. Make the script executable (as root):

   ```
   chmod 755 /etc/rc.d/init.d/<pf_user>
   ```

8. Test the script by entering:

   ```
   service <pf_user> start
   ```

   and then:

   ```
   service <pf_user> stop
   ```

9. To start the service, enter:

   ```
   service <pf_user> start
   ```

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**.

   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. Log on as the root user.
2. Stop the service with the command:

   ```
   service <pf_user> stop
   ```

   where `<pf_user>` is the PingFederate service user account (see *Install the PingFederate service on Linux manually*).
3. Remove symbolic links:

   ```
   rm /etc/rc3.d/S84<pf_user>
   rm /etc/rc4.d/S84<pf_user>
   rm /etc/rc5.d/S84<pf_user>
   rm /etc/rc0.d/K15<pf_user>
   rm /etc/rc1.d/K15<pf_user>
   rm /etc/rc2.d/K15<pf_user>
   rm /etc/rc6.d/K15<pf_user>
   ```
4. Optional: Delete the script used to start and stop the service.
5. Optional: Remove the PingFederate installation directory (`pf_install`).

The administrative console

The administrator's user interface for PingFederate®, the *administrative console*, is built around a system of wizard-like control screens. To access the administrative console, launch a web browser and go to:

```
https://<DNS_NAME>:9999/pingfederate/app
```

where `<DNS_NAME>` is the fully qualified name of the machine running the PingFederate server.

Once logged on, the Main menu offers up to four menu choices:

- IdP Configuration
- SP Configuration

- OAuth Settings
- Server Configuration

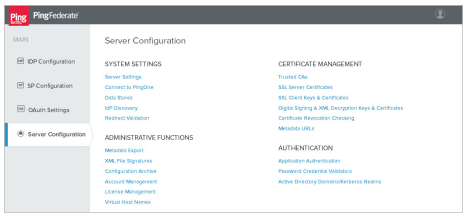


Figure 4: A sample of the Server Configuration menu

Note that the menu choices and menu items varies with the federation roles PingFederate plays; for examples, IdP, SP, SCIM provisioner, or OAuth Authorization Server (see *Choose roles and protocols* on page 136 in the "System Settings" chapter of the PingFederate *Administrator's manual*). Menu items also depend on the permissions assigned to the logged-on administrator (see *Account management* on page 105 in the "System Administration" chapter of the PingFederate *Administrator's manual*).

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.

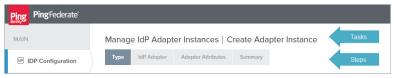


Figure 5: Sample tasks and steps

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.

Button	Description
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 all versions of the Security Assertion Markup Language (SAML) protocol, from 1.0 through 2.0, OAuth, and for WS-Trust, which underlies the PingFederate STS for Web Services. In addition, PingFederate® supports the WS-Federation browser-based, "passive" protocol using SAML assertions as SSO-enabling security tokens.

This chapter describes:

- *Federation roles* on page 28
- *Terminology* on page 29
- *SAML 1.x profiles* on page 30
- *SAML 2.0 profiles* on page 33
- *WS-Federation* on page 43
- *About account linking* on page 44
- *Web Services standards* on page 45
- *OAuth 2.0* on page 47
- *System for Cross-domain Identity Management (SCIM)* on page 50
- *Transport and message security* on page 50

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 specifications and available in PingFederate, is that of an IdP Discovery provider.

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 this manual.

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 a SAML assertion.

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 (see *Standard IdP Discovery* on page 42).

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 30.

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. (See *Export metadata to an XML file* on page 101 in the “System Administration” chapter of the PingFederate *Administrator's manual*.).

Whether you are exporting or importing a metadata file, PingFederate supports the use of XML digital signatures to ensure the integrity of the data (see *Sign XML files* on page 103 in the “System Administration” chapter of the PingFederate *Administrator's manual*.).

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, for example, *oasis-sstc-saml-core-1.1.pdf*).

An administrator can configure PingFederate, acting as an IdP, to include authentication context in assertions. For information about this configuration, see *Create an attribute contract* on page 259 in the “Identity Provider SSO Configuration” chapter of the PingFederate *Administrator's manual* or *Define an attribute contract for IdP STS* on page 388 in the WS-Trust STS Configuration chapter.

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 (see *SSO integration kits and adapters* on page 64 in the “Key Concepts” chapter of the PingFederate *Administrator's manual*). Conversely, the SP developer can call methods for extracting authentication context from an assertion. It is up to the SP developer and application to create access control or other processing based on the context.

Check the *User Guide* for your integration kit to see if this feature is supported.

For more information on configuring authentication context for an adapter instance, see *Select an authentication context* in the “Identity Provider SSO Configuration” chapter of the PingFederate *Administrator's manual*.

Browser-based SSO

Browser-based SSO includes SAML 1.x, 2.0, and WS-Federation and provides standards-based SSO, Single Logout, 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.

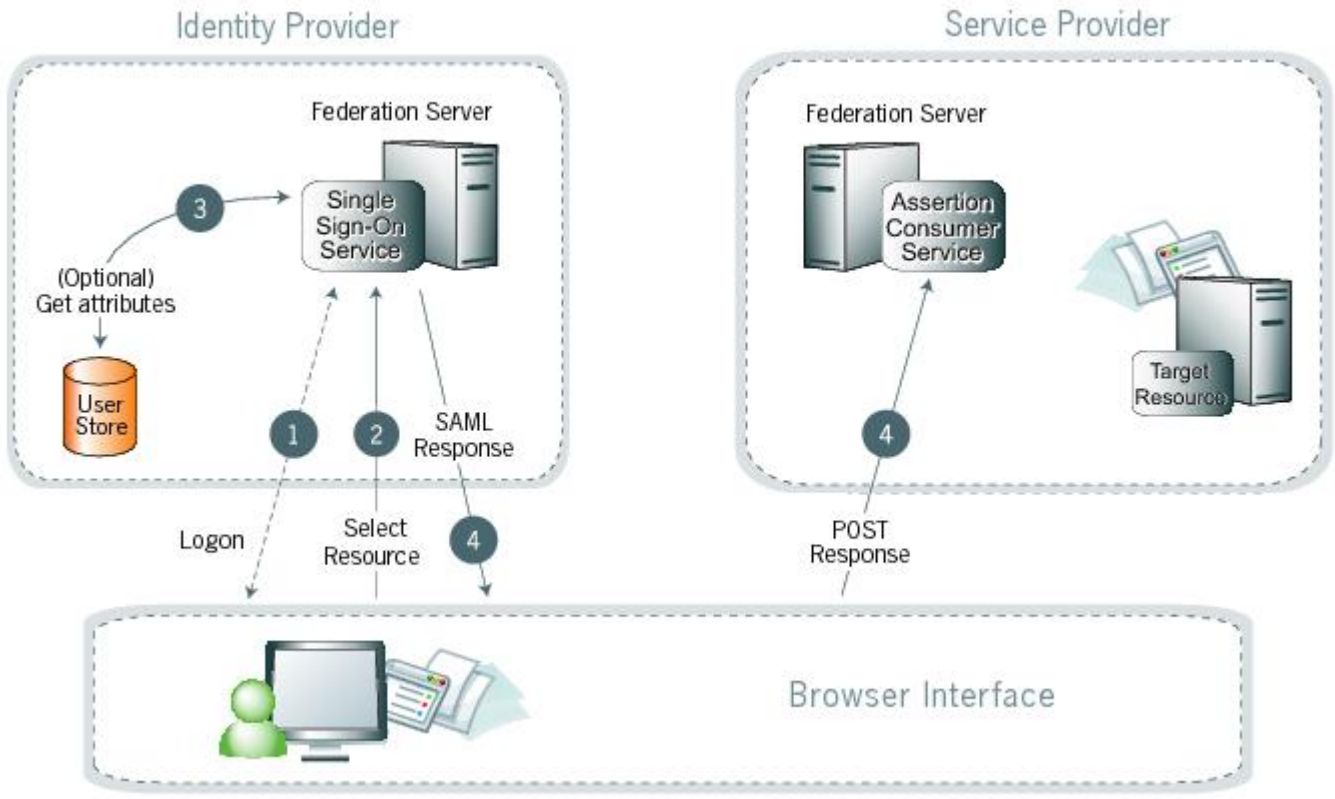


Figure 6: Browser/POST Profile

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.

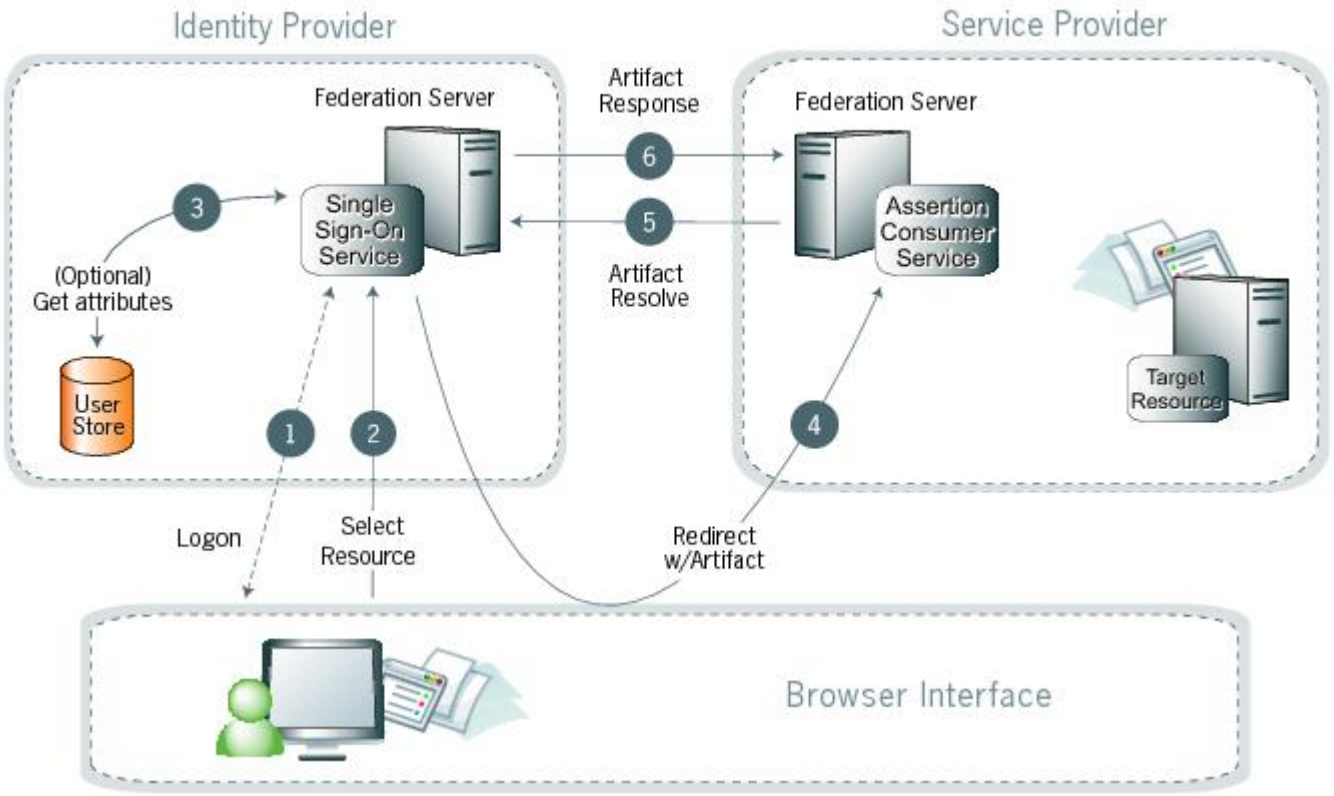


Figure 7: SSO:Browser/Artifact Profile

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 (a.k.a. “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.

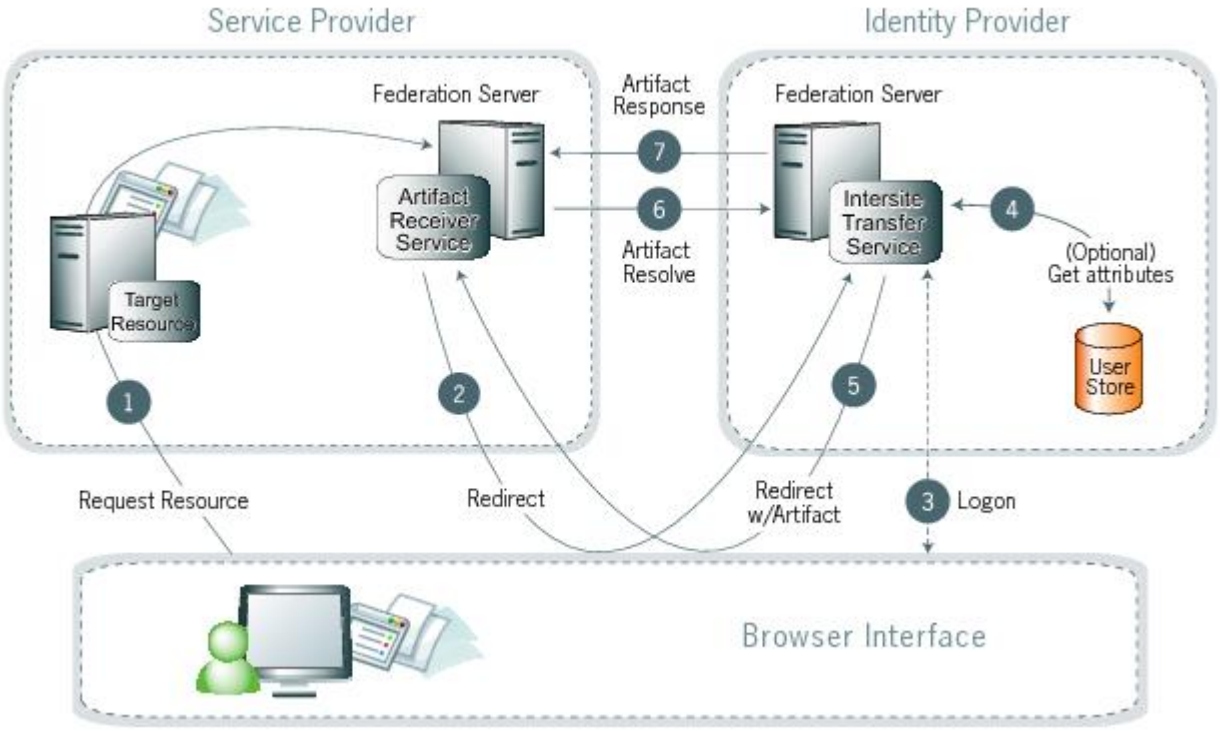


Figure 8: SP-Initiated SSO

Processing Steps:

1. The user requests access to a protected SP resource. The request is redirected to the federation server (e.g., 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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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 34
- *Single logout* on page 41
- *Attribute Query and XASP* on page 42
- *Standard IdP Discovery* on page 42

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 34
- *SP-initiated SSO—Redirect-POST* on page 35
- *SP-initiated SSO—Artifact-POST* on page 36
- *SP-initiated SSO—POST-Artifact* on page 37
- *SP-initiated SSO—Artifact-Artifact* on page 39
- *SP-initiated SSO—Redirect-Artifact* on page 38
- *IdP-initiated SSO—POST* on page 40
- *IdP-initiated SSO—Artifact* on page 41

SP-initiated SSO—POST-POST

In this scenario a user attempts to access a protected resource directly on an SP web site 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.

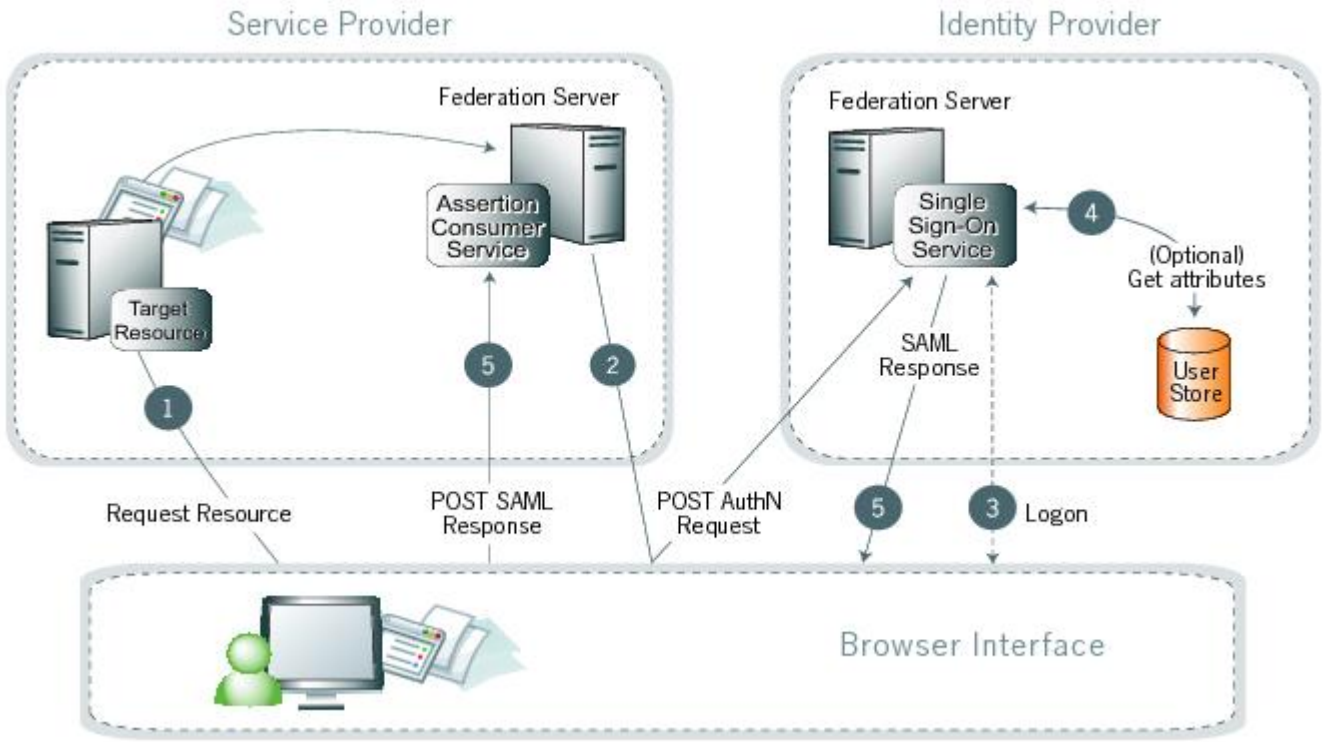


Figure 9: SP-Initiated SSO: POST/POST

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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.

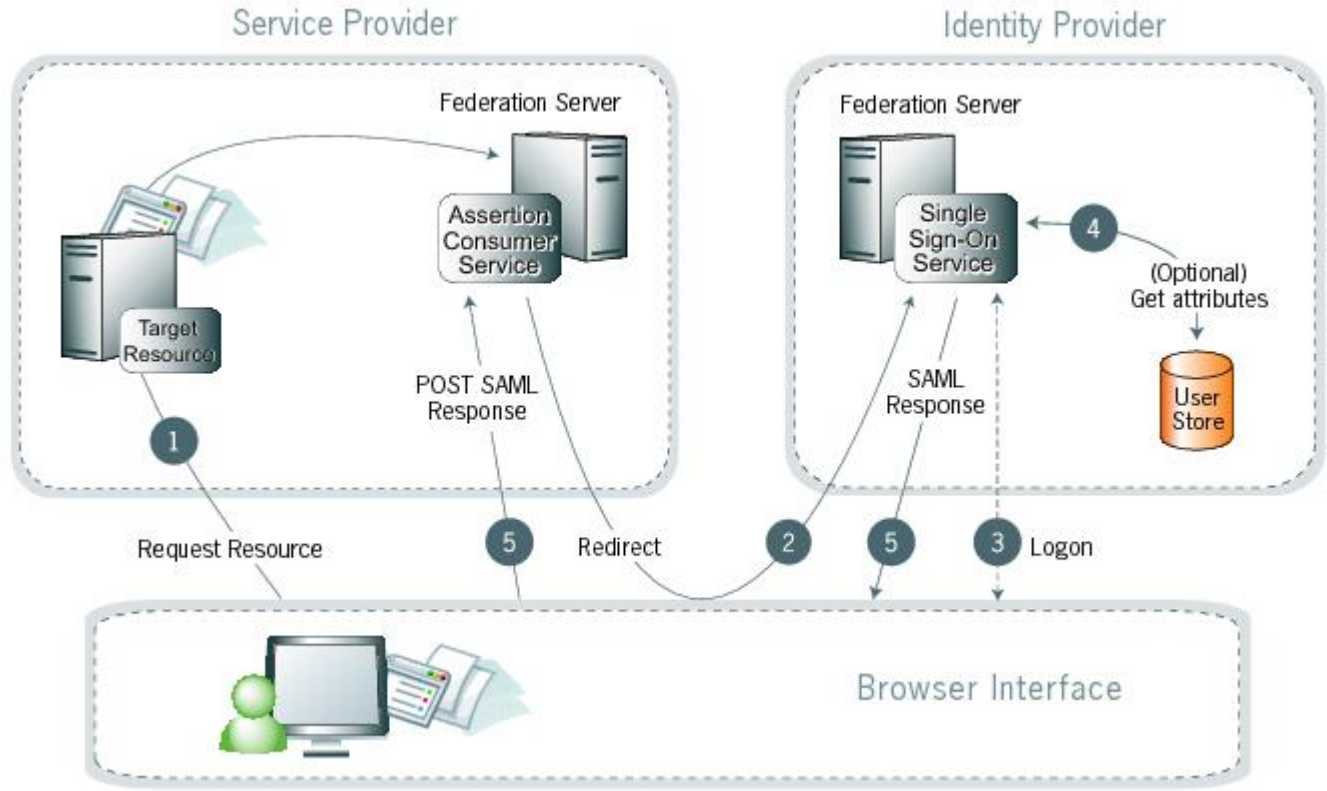


Figure 10: SP-Initiated SSO: Redirect/POST

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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.

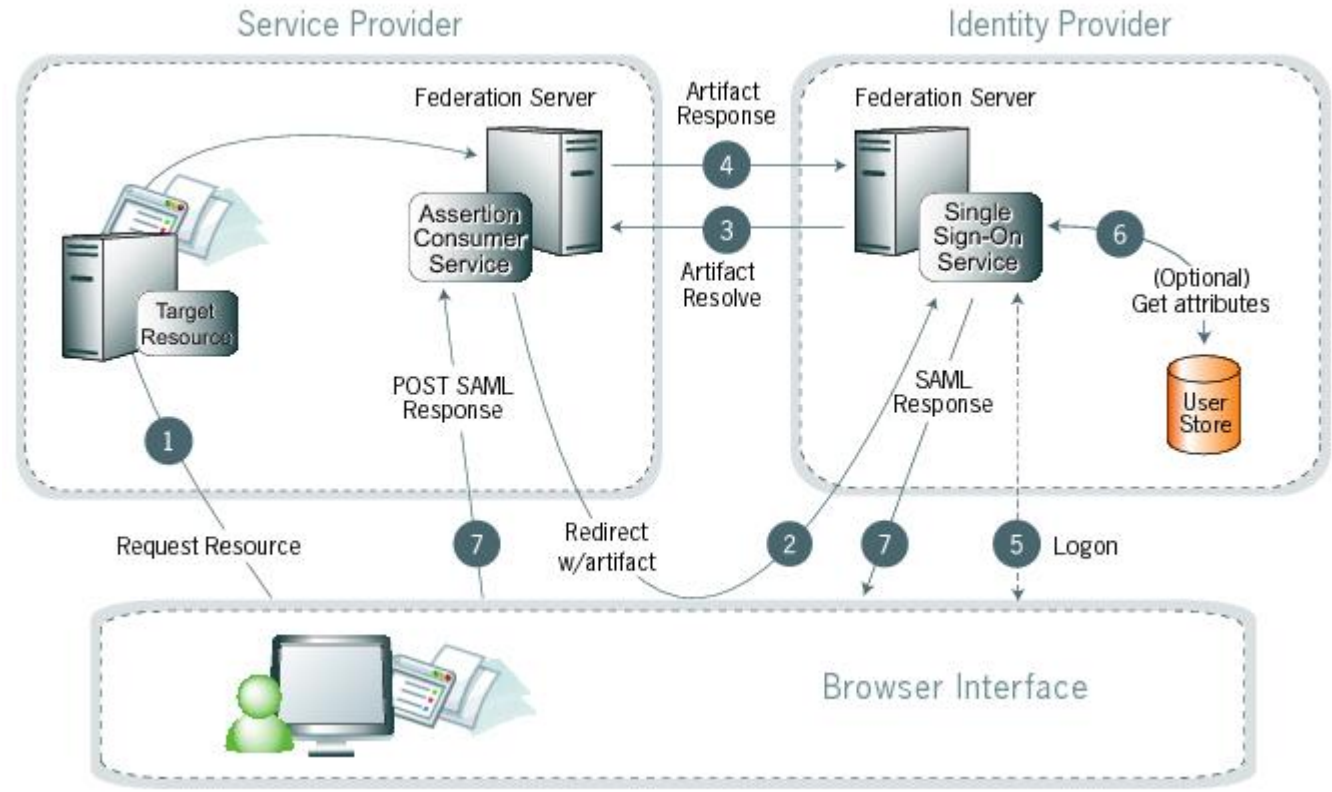


Figure 11: SP-Initiated SSO: Artifact/POST

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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 .

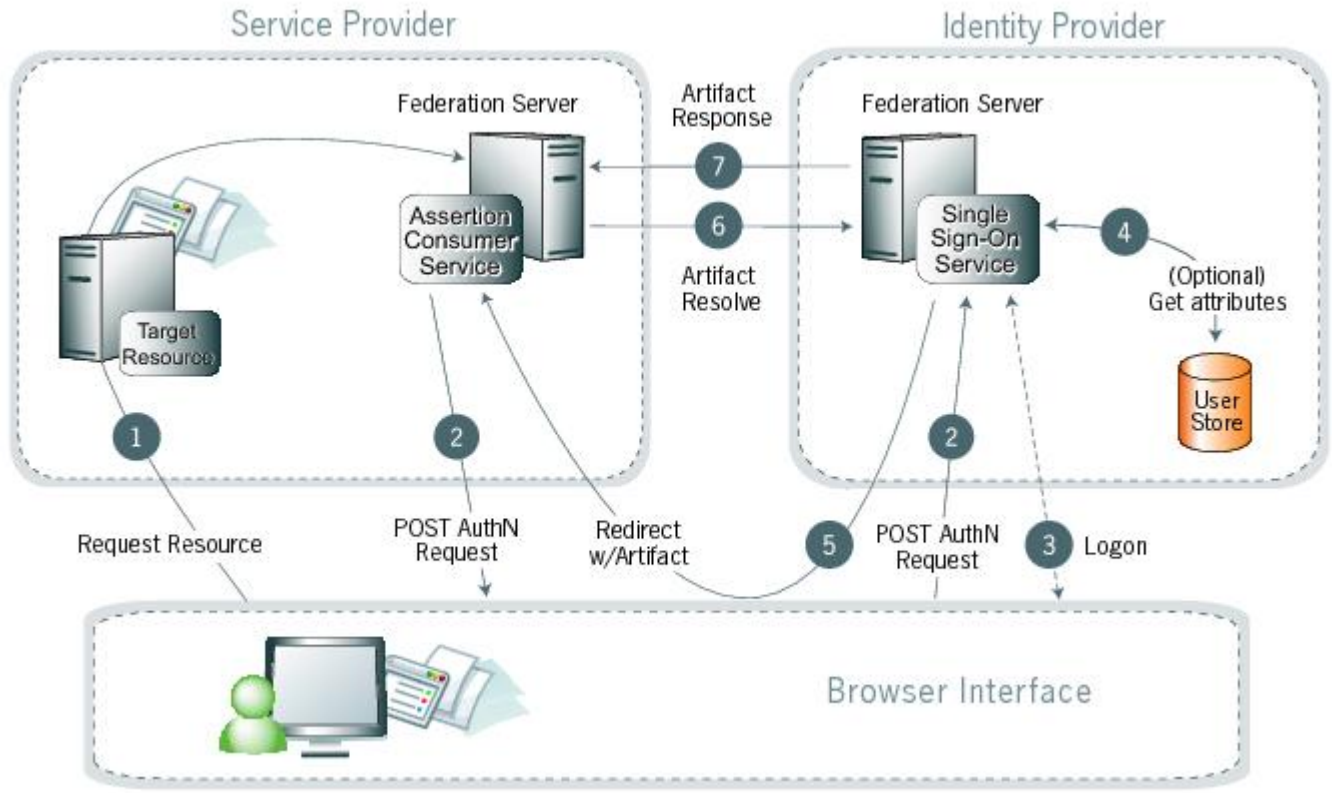


Figure 12: SP-Initiated SSO: POST/Artifact

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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.

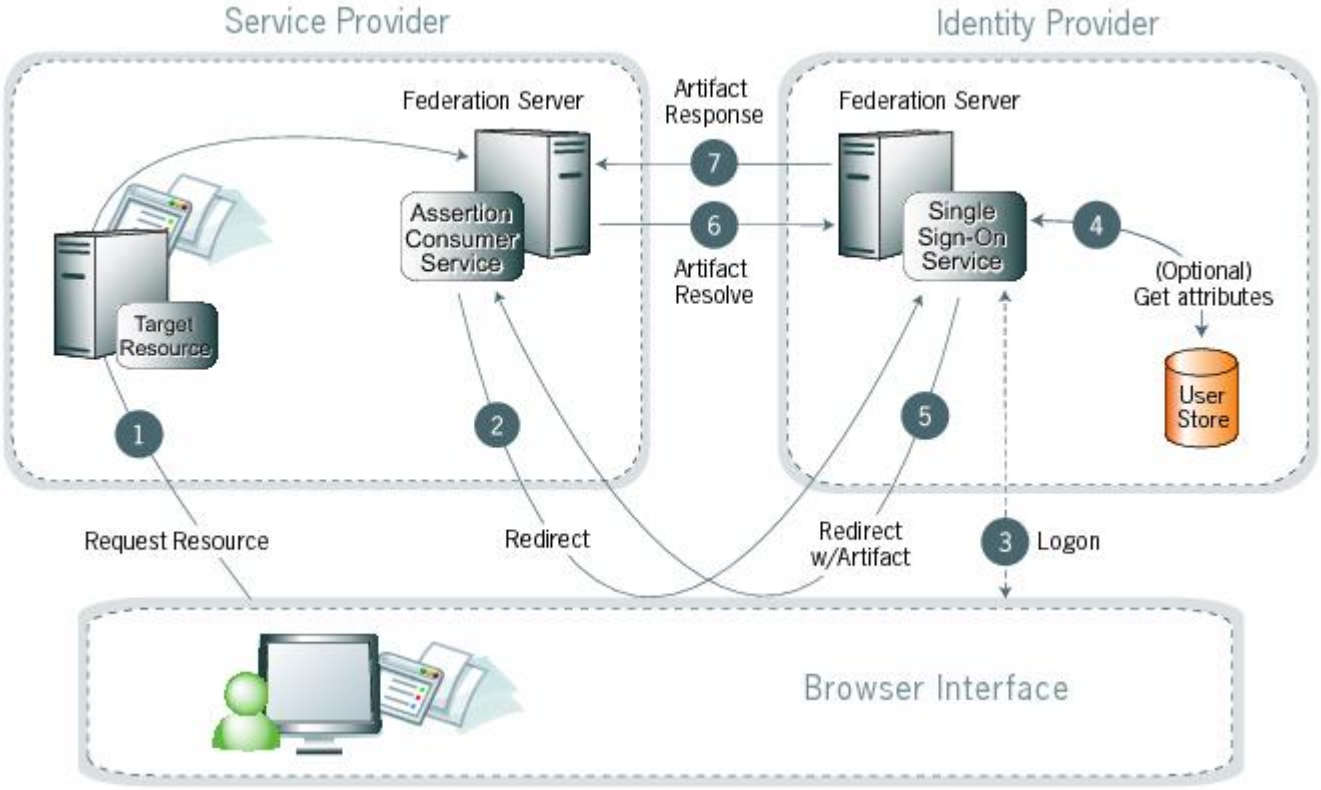


Figure 13: SP-Initiated SSO: Redirect/Artifact

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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.

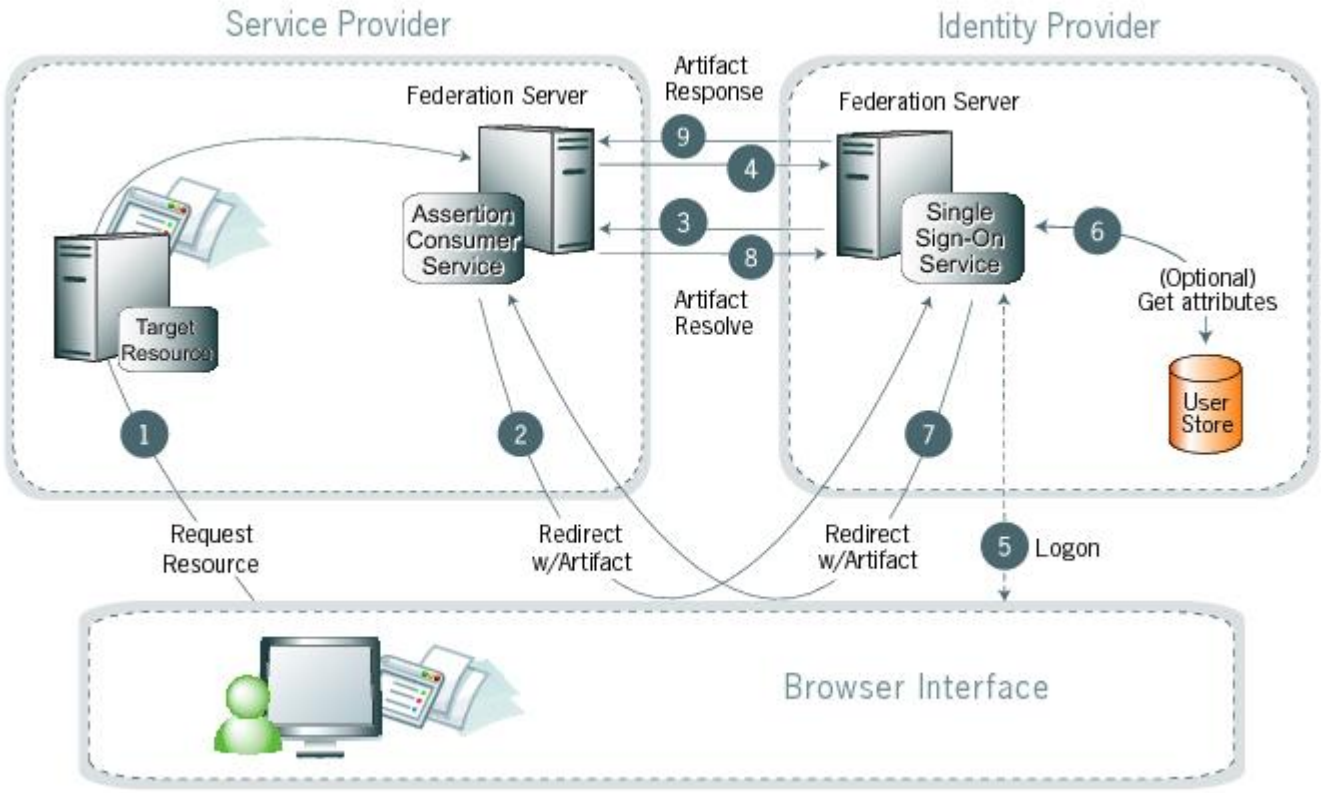


Figure 14: SP-Initiated SSO: Artifact/Artifact

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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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.

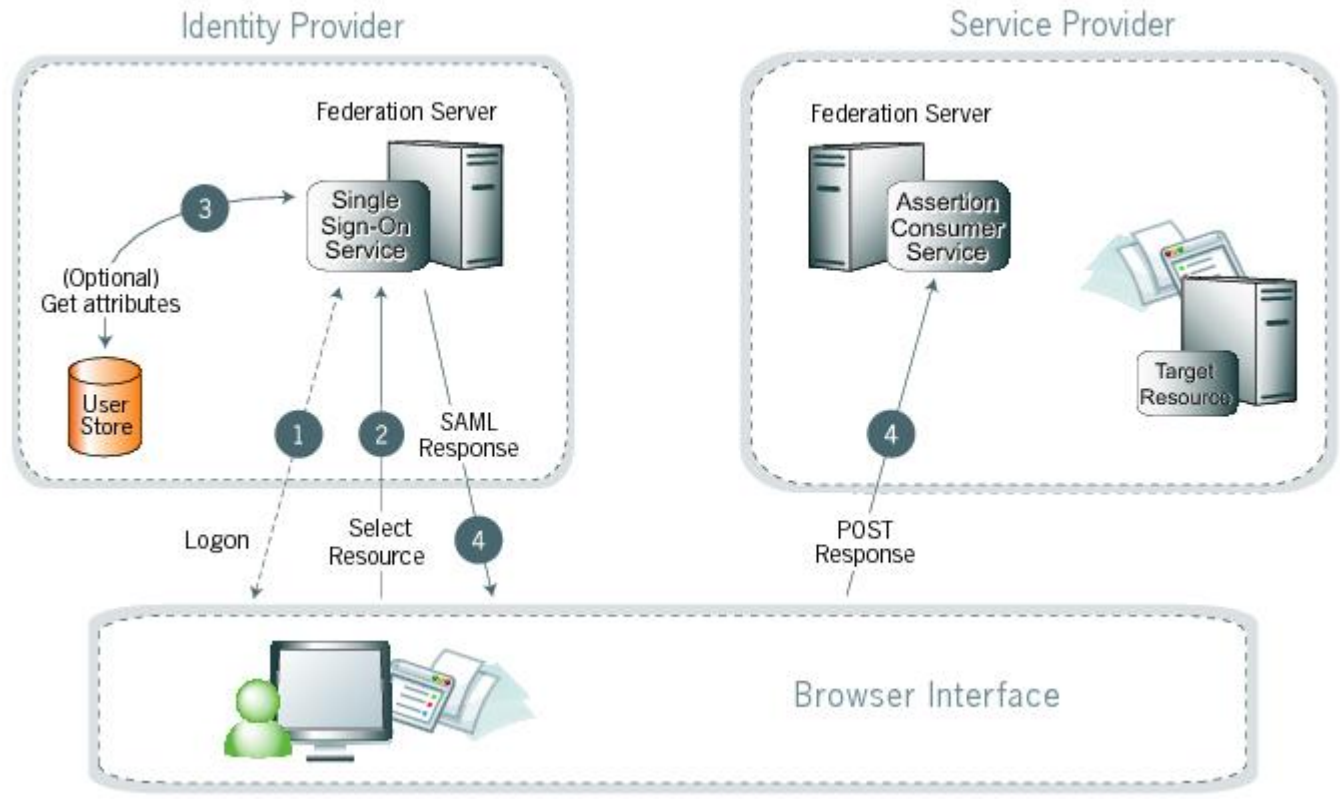


Figure 15: IdP-Initiated SSO: POST

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.

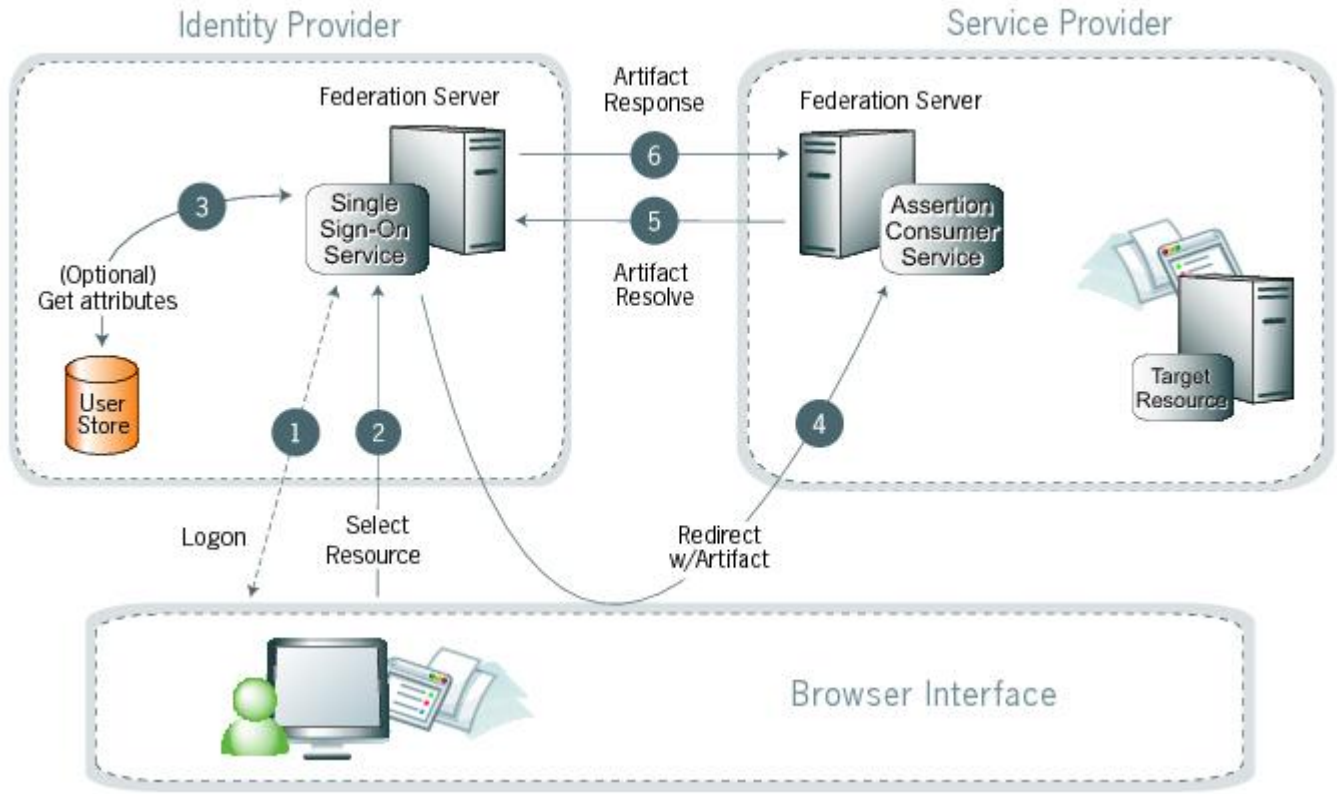


Figure 16: IdP-Initiated SSO: Artifact

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 *Single sign-on* on page 34 for illustrations of these message flows.

About session clean-up

When an SP receives an SLO request from an IdP, the session creation adapter(s) you are using must handle any session clean-up with respect to the local application. For more information about adapters, see *SSO integration kits and adapters* on page 64 in the Key Concepts chapter of the PingFederate *Administrator's manual*.

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 (see *Attribute masking* on page 71 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*).

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 (see *Authentication* on page 172 in the "Security Management" chapter of the PingFederate *Administrator's manual*). 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 (see *Manage attribute requester mappings* on page 316 in the "Service Provider SSO Configuration" chapter of the PingFederate *Administrator's manual*). 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 (see *SSO integration kits and adapters* on page 64 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*).

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 156 in the "System Settings" chapter of the PingFederate *Administrator's manual*.

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.

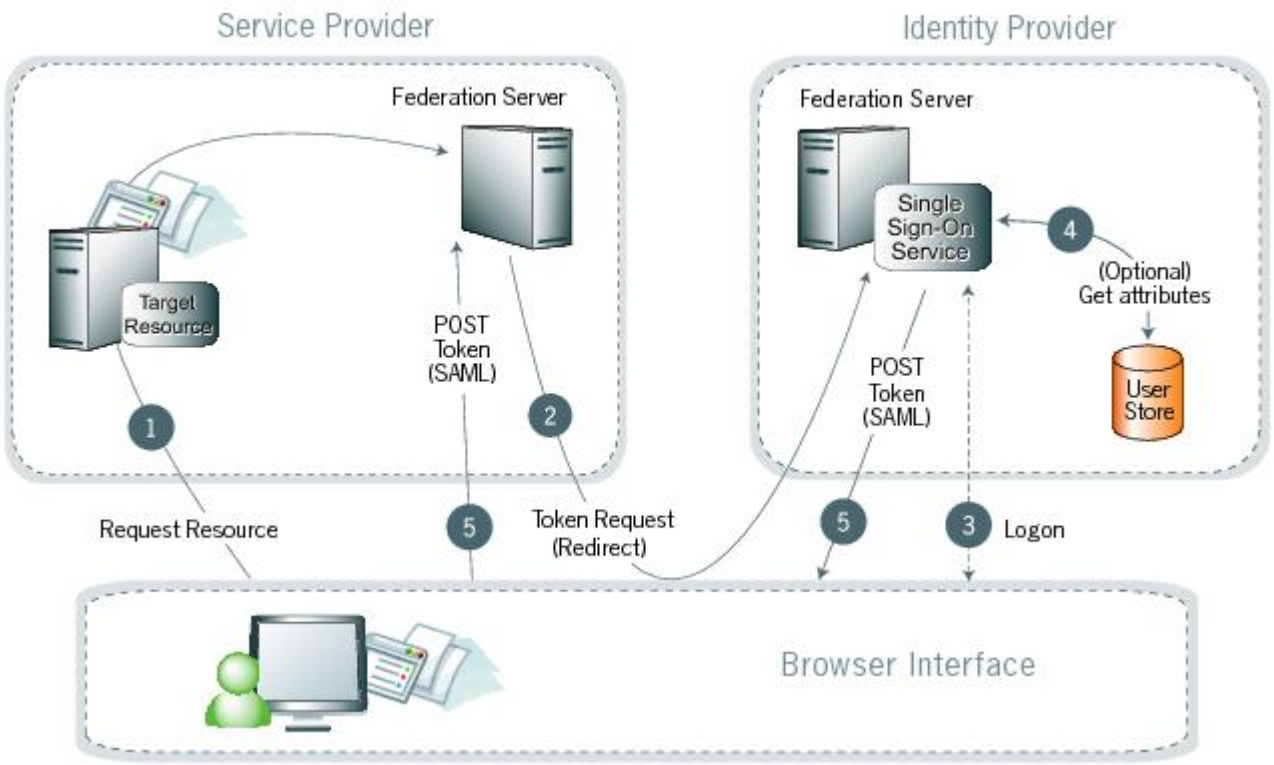


Figure 17: WS-Federation SSO

Processing Steps:

1. A user requests access to a protected SP resource. The user is not logged on to the site. The request is redirected to the federation server to handle authentication.
2. The SP generates a security token request and redirects the browser to the identity provider's WS-Federation implementation.
3. If the user is not already logged on to the IdP site or if re-authentication is required, the IdP asks for credentials (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 *About attributes* on page 67 in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.)
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 41); 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.

For more information about account linking, see *Account linking* in the "Key Concepts" chapter of the PingFederate *Administrator's manual*.

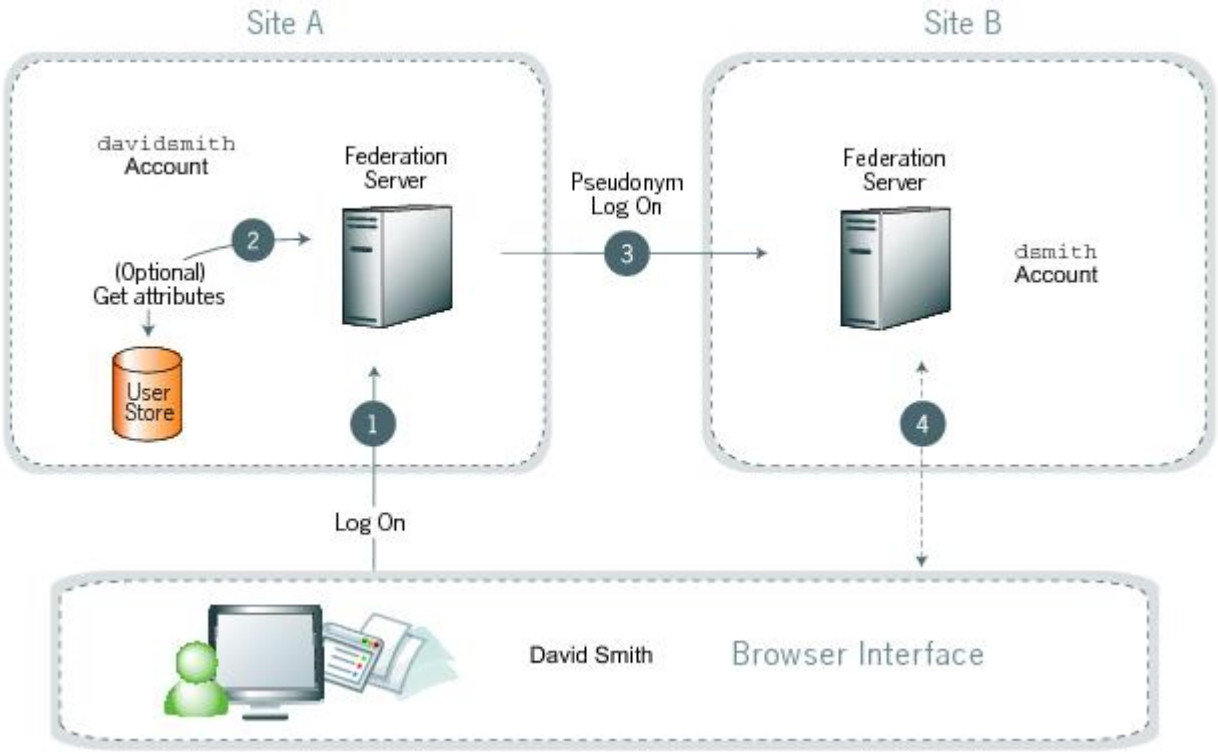


Figure 18: Account Linking

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(see Figure 21). 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 46 below).

Figure 19: 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 *Figure 20: Token Exchange (Example)* on page 46).

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 *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-sx*.)

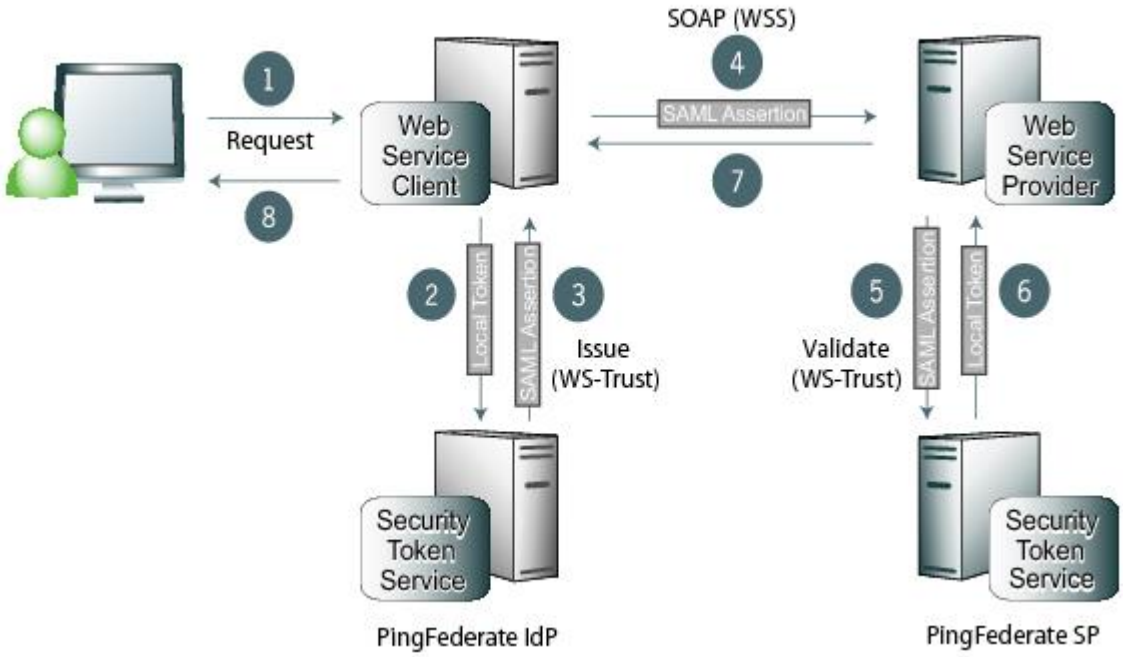
Request types

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 20: 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 Client and Provider sides of the interaction. However, other deployment options are also supported.

OAuth 2.0

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

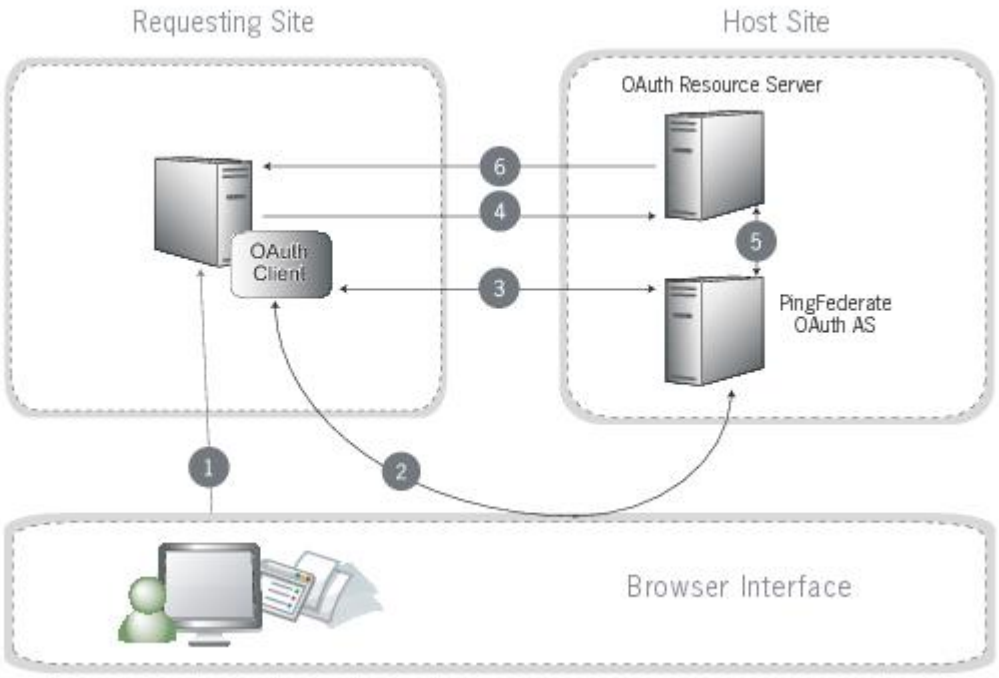
- **Access Token** – Allows clients to authenticate to a resource server and claim authorizations for accessing particular resources. Access tokens have specific authorization scope and duration.
- **Refresh Token** – Allows clients to obtain a fresh access token without re-obtaining authorization from the resource owner. It is a long-lived token that a client can trade in to an authorization server to obtain a new 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.

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.



Processing Steps

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

   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 an information access 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.

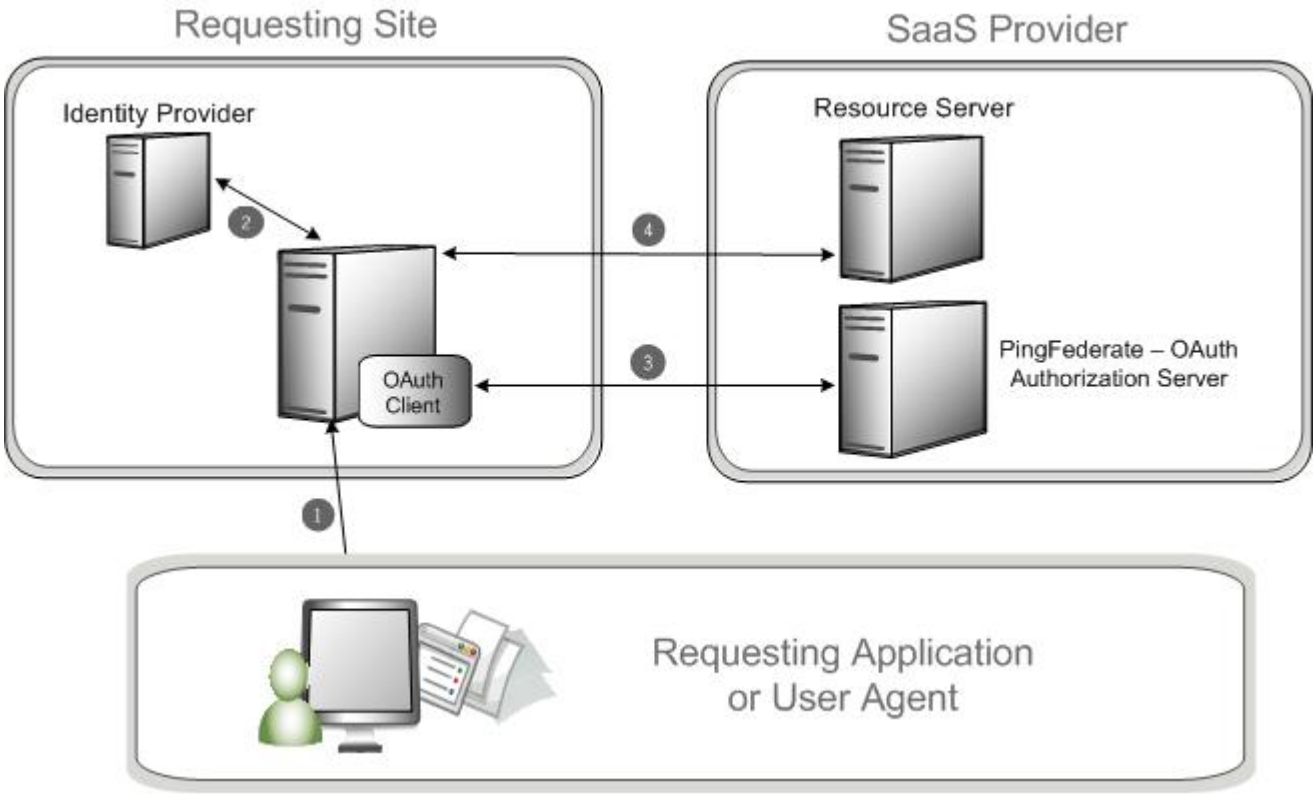
3. The requesting site makes an HTTPS request to the OAuth AS to exchange the authorization code for an access token.

   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 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 Resource Server returns the requested data to the requesting site.
7. (Not shown.) The requesting site displays data from the API call to the user.

SAML 2.0 profile for OAuth 2.0 authorization grants

In this scenario, a client obtains a SAML 2.0 bearer assertion and makes an HTTP request to the PingFederate® OAuth AS to exchange the SAML 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 to obtain data.



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 a SAML 2.0 bearer assertion from a local Identity Provider (IdP); for example, the PingFederate IdP server.

3. The client application makes an HTTP request to the PingFederate OAuth AS to exchange the SAML assertion for an access token. The OAuth AS validates the assertion and returns the access token. For more information on how the OAuth AS performs the validation, see the *specification* (tools.ietf.org/html/draft-ietf-oauth-saml2-bearer).
4. The client application adds the access token to its API call to the Resource Server. The Resource Server returns the requested data to the client application.

OpenID Connect support

As an extension of OAuth, PingFederate also supports Basic and Implicit Profiles defined for OpenID Connect, an emerging standard. For more information, see the *OpenID Connect web site* (`openid.net/connect`).

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 client credentials	HTTP Basic and client certificate (mutual TLS)
Source data stores	Active Directory and other LDAPv3-compliant directory servers	Not appliable
Target data stores	Not applicable	Active Directory and custom data stores (via the Identity Store Provisioner Java SDK interface)

For more information regarding outbound provisioning for IdPs and inbound provisioning for SPs, see *User provisioning* on page 78 in the “Key Concepts” chapter of the PingFederate *Administrator's manual*. For detailed information about SCIM, see the web site *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* available on the *SSTC web site* (`www.oasis-open.org/committees/tc_home.php?wg_abbrev=security#samlv20`).

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:

- 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, use the information in the following sections to configure PingFederate to interact with the HSM for key generation, storage, and operation.

If you have an existing PingFederate environment, follow the vendor-specific instructions to install and configure your HSM and a *new* PingFederate installation, and then mirror the configuration based on the existing environment.

Install and configure Luna SA client and PingFederate

1. Install and configure your 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 following files to the Java installation:

   For Windows:

   - Copy the `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 following files from the `LUNA_HOME\jsp\lib` directory to the `JAVA_HOME\jre\lib\ext` directory:

     Luna 4.x: `LunaJCASP.jar` and `LunaJCESP.jar`

     Luna 5.x: `LunaProvider.jar`

   For UNIX/Linux:

   - Copy the following files from the `LUNA_HOME/jsp/lib` directory to the `JAVA_HOME/jre/lib/ext` directory:

     `libLunaAPI.so`

     Luna 4.x: `LunaJCASP.jar` and `LunaJCESP.jar`

     Luna 5.x: `LunaProvider.jar`

   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 support for more information.
3. In the Java SDK directory, edit the `java.security` file in the `jre/lib/security` directory and add the line (or lines) in **bold** to the list of security providers, *immediately before* the `sun.security.ec.SunEC` providers:

Luna 4.x:

```
# 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.chrysalisits.crypto.LunaJCAProvider
security.provider.4=com.chrysalisits.cryptox.LunaJCEProvider
security.provider.5=sun.security.ec.SunEC
security.provider.6=com.sun.net.ssl.internal.ssl.Provider
security.provider.7=com.sun.crypto.provider.SunJCE
security.provider.8=sun.security.jgss.SunProvider
security.provider.9=com.sun.security.sasl.Provider
security.provider.10=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.11=sun.security.smartcardio.SunPCSC
```

Luna 5.x:

```
# 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. Install PingFederate on the network interconnected to the HSM (see *Installation*).
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:

      Luna 4.x:: `<construct class=`**`"com.pingidentity.crypto.LunaJCEManager"`**`/>`

      Luna 5.x:: `<construct class=`**`"com.pingidentity.crypto.LunaJCEManager5"`**`/>`
   b) For the `CertificateService` service endpoint, change the value of the construct class as follows:

      Luna 4.x: `<construct class=`**`"com.pingidentity.crypto.LunaCertificateServiceImpl"`**`/>`

      Luna 5.x: `<construct class=`**`"com.pingidentity.crypto.LunaCertificateServiceImpl5"`**`/>`
6. Edit the `run.properties` file in the `<pf_install>/pingfederate/bin` directory and change the value of the pf.hsm.mode property from `OFF` to `LUNA`; for example:

   `pf.hsm.mode=`**`LUNA`**
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 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`

   This completes the steps required to configure PingFederate for use with the Luna SA. You may start the PingFederate server in the normal way and proceed as you would for any other installation.

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.

Luna SA operational notes

Some restrictions apply to PingFederate operations when using an HSM:

Note: PingFederate does not store public certificates (for both signature and encryption) on the hardware module. In this case, certificates are stored in keystores located on the file system.

- As an OpenID Connect Provider, PingFederate generates and rotates temporary asymmetric key pairs to sign ID Tokens for Relying Parties. These in-memory short-term keys are not stored on the 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.
- 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.
- 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.
- If you have an existing PingFederate environment, create a new PingFederate installation, follow the steps in *Install and configure Luna SA client and PingFederate* on page 51 to integrate the new installation with Luna SA, and mirror the configuration based on the existing environment.

Install and configure nShield Connect client and PingFederate

1. Install and configure your 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 currently supports only 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 `nCipherKM.jar` file from the `NFAST_HOME\java\classes` directory 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. In your Java SDK directory, edit the `java.security` file in the `jre/lib/security` directory and add the line in **bold** to the list of security providers, *after* all the Sun providers:

```
# 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. Install PingFederate on the network interconnected to the HSM (see *Installation*).
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 `run.properties` file in the `<pf_install>/pingfederate/bin` directory and change the value of the pf.hsm.mode property from `OFF` to `NCIPHER`; for example:

   ```
   pf.hsm.mode=NCIPHER
   ```

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. For clustered-server installations, see *Additional steps for server clusters* on page 54.

Additional steps for server clusters

If your PingFederate installation is configured in a clustered environment, use these steps to replicate nShield data to other connected nodes in the cluster.

1. On the console node, locate the `<pf_install>/pingfederate/server/default/data` directory and create a sub directory named `ncipher-kmdata-local`.
2. Copy to the `ncipher-kmdata-local` directory all files from the `NFAST_KMDATA\local` directory, where `NFAST_KMDATA` is an environment variable created during the nShield Connect installation.

   For example, `NFAST_KMDATA` could be set to `C:\ProgramData\nCipher\Key Management Data`.
3. Create a new environment variable named `NFAST_KMLOCAL` and set it to `<pf_install>/pingfederate/server/default/data/ncipher-kmdata-local`

   Note: Perform this step on all servers within the cluster.
4. Restart the nShield Connect hardserver on all PingFederate servers in the cluster. (See the Thales documentation for instructions on restarting the hardserver.)
5. 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 to the engine nodes.

nShield Connect operational notes

Some restrictions apply to PingFederate operations when using an HSM:

Note: PingFederate does not store public certificates (for both signature and encryption) on the hardware module. In this case, certificates are stored in keystores located on the file system.

- As an OpenID Connect Provider, PingFederate generates and rotates temporary asymmetric key pairs to sign ID Tokens for Relying Parties. These in-memory short-term keys are not stored on the 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.
- 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.
- 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.
- If you have an existing PingFederate environment, create a new PingFederate installation, follow the steps in *Install and configure nShield Connect client and PingFederate* on page 53 to integrate the new installation with Thales nShield Connect, and mirror the configuration based on the existing environment.

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 56 — 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 87 — Information about maintaining the PingFederate server and deployment, using log files, managing users, and handling other administrative functions.
- *System settings* on page 129 — How to configure your local PingFederate server settings.
- *OAuth configuration* on page 181 — How to configure PingFederate to act as an OAuth Authorization Server.
- *Security management* on page 160— Information about importing, exporting, and maintaining certificates and keys in PingFederate.
- *Authentication policies* on page 208— Configure Authentication Policies, Selectors, and Policy Contracts to fulfill various authentication requirements, to deploy PingFederate as a Federation Hub, or both.
- *Identity provider SSO configuration* on page 238— How to configure PingFederate to act as an Identity Provider (IdP) and establish connections to Service Providers.
- *Service provider SSO configuration* on page 308 — How to configure PingFederate to act as a Service Provider (SP) and establish connections to Identity Providers.
- *WS-Trust STS configuration* on page 378 — 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 417 — Configure advanced federation use cases, such as Adapter-to-Adapter Mappings (for Browser SSO) and Token Exchange Mappings (for STS).
- *Kerberos Adapter* on page 424 — How to configure PingFederate to use the packaged Kerberos Adapter.
- *HTML Form Adapter* on page 428 — How to configure PingFederate to use the packaged HTML Form Adapter.
- *HTTP Basic Adapter* on page 432 — How to configure PingFederate to use the packaged HTTP Basic Adapter.
- *OpenToken Adapter* on page 434 — How to configure PingFederate to use the packaged OpenToken Adapter for interfacing with your web applications.
- *Composite Adapter* on page 439 — How to configure PingFederate to use the packaged Composite Adapter.
- *Application endpoints* on page 442 — Detailed information about using PingFederate connection endpoints for web single sign-on and single logout.
- *OAuth 2.0 endpoints* on page 455 — Detailed information about using PingFederate connection endpoints for the OAuth.
- *Web Service interfaces* on page 465 — Information developers can use to automate connection-configuration management and runtime SSO partner discovery.
- *Attribute mapping expressions* on page 485 — How to enable and use expressions in conjunction with mapping attributes.
- *Customize assertions and authentication requests* on page 489 — Information about assertion and authentication request customizations.
- *Fulfillment by data store queries* on page 494 — 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 500 — Solutions for difficulties that may be encountered.
- *Glossary* on page 505 — Definitions of terms used in the manual and in identity federation parlance.
- *List of acronyms* on page 513 — A list of acronyms.

Key concepts

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

- *Connection types* on page 57
- *About WS-Trust STS* on page 57
- *PingFederate OAuth AS* on page 60
- *SSO integration kits and adapters* on page 64
- *Hierarchical plug-in configurations* on page 66
- *Identity mapping* on page 66
- *About attributes* on page 67
- *Security infrastructure* on page 72
- *Using Auto-Connect* on page 75
- *User provisioning* on page 78
- *Federation planning checklist* on page 79

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 7 in the *Get started with PingFederate* guide.

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 28 in the *Get started with PingFederate* guide).
- WS-Trust STS – Employs the PingFederate Security Token Service (STS), which enables Web Service Client and Provider applications 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 57). These standards, including WS-Trust, do not rely on the user's browser for message transport.
- OAuth SAML Grant – Exchanges a SAML assertion for an OAuth access token with the PingFederate Authorization Server (AS) (see *PingFederate OAuth AS* on page 60).
- Provisioning – Provides automated cross-domain inbound and outbound user management (see *User provisioning* on page 78).

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 45 in the "Supported Standards" chapter of the *Get started with PingFederate* guide.)

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 58).

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 45 in the "Supported Standards" chapter of the *Get started with PingFederate* guide).

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 64).

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 421).

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 66).

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 60). (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 web site (`www.pingidentity.com/en/products/downloads.html`).

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 web site* (`www.pingidentity.com/en/products/downloads.html`).

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 69).

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* (`msdn.microsoft.com/en-us/library/ms730879.aspx`).

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:

- Federation Metadata, which contains details on the PingFederate public signing certificate and other information required to establish the trust relationship (see */pf/federation_metadata.ping* on page 453).
- Metadata Exchange, which contains connection details relating to the SP partner (see */pf/sts_mex.ping* on page 453).

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

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 (Web Service Client) 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* specification (tools.ietf.org/html/draft-ietf-oauth-saml2-bearer). This capability provides a bridge between 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.

OAuth Access Token via SAML 2.0 Bearer Assertion

`oauth-v2:access:token:response|via|urn:ietf:params:oauth:grant-type:saml2-bearer`

This proprietary token request is similar to the one above 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 element `<AppliesTo>` of the WS-Trust RST message.)

PingFederate OAuth AS

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 to the RS (for example, as a parameter on the API call) 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 – 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 – Serve as a reference to some set of attributes. The RS must de-reference the token for the corresponding identity and security information at the OAuth AS that issued it.

Token management

PingFederate® supports multiple access token management instances, providing flexibility for enterprises where deployments may require different token data models, token lifetimes, attribute contracts, token validation rules, or any combination of them, for various clients.

Grant types

To obtain an access token, a client interacts with an OAuth 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 (and often a refresh) token. Resource owner credentials are never exposed to the client.

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.

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 tokens to exist between the client and the Resource Server, and long-lived tokens between the client and the AS.

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.

Persistent vs. transient grants

There are two types of OAuth grants, namely persistent grants and transient 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 are valid until they are explicitly revoked. Persistent grants can result from the following scenarios:

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

Persistent grants support extended attributes, making it possible to map attribute values obtained during a user's initial authentication into the access tokens. Extended attributes are optional and managed as part of the OAuth AS settings.

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`, 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 once a day. For database storage, you can fine-tune the frequency and the number of expired grants (per batch) to be removed.

Important: PingFederate uses a pre-installed HyperSQL 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.

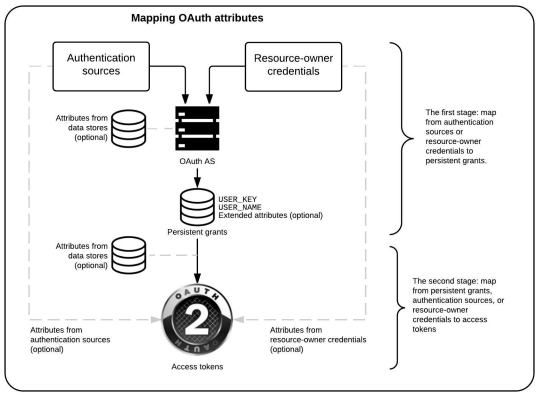
Tip: The Javadoc for PingFederate is located the `<pf_install>/pingfederate/sdk/doc` directory.

Mapping OAuth attributes

Mapping OAuth attributes is a two-stage processing workflows:

- The first stage: map from authentication sources or resource-owner credentials to persistent grants.
- The second stage: map from persistent grants, authentication sources, or resource-owner credentials 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.



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—via SAML assertions or WS-Federation assertions via IdP connections, IdP adapters, resource owner credentials, of any combination of these use cases. Data-store lookup is provided as needed (for example, to retrieve an LDAP ID for storage as the user key).

Note: The `USER_KEY` values must be unique across all end users because it is the identifier to store and to retrieve persistent grants. For example, if you have two Active Directory domains, the sAMAccountName 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 Subject DN to the `USER_KEY`.

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.

If the authentication context matches a specific mapping, then attributes from the authentication sources or resource owner credentials can also be used to map into OAuth 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.

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 these approved scopes. Scopes are configured globally in the **OAuth Settings** > **Scope Management** screen but may be restricted on a client-by-client basis.

OAuth user-facing screens

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.

Tip: These screens can be customized and branded as needed (see *Customizable user-facing screens* on page 120).

OpenID Connect

As an extension of OAuth capabilities, PingFederate® supports an optional configuration for OpenID Connect, an emerging protocol for secure, lightweight transfer of authentication and user attributes (see *openid.net/connect*).

PingFederate supports both the OpenID Connect Basic and Implicit Profiles defined in the standard. In both of these 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 OpenID Connect Provider (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`).

To configure OpenID Connect policies, first enable the protocol in **Server Configuration** > **Server Settings** > **Roles & Protocols** screen. To make use of OpenID Connect defined scopes for accessing various degrees of user-associated claims, you must also configure them in the **OAuth Settings** > **Scope Management** screen.

Tip: For a list of OpenID Connect defined scopes, see *5.4. Requesting Claims using Scope Values* in the specification (openid.net/specs/openid-connect-core-1_0.html#ScopeClaims).

Furthermore, 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`).

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 531 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 65).

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 424).

HTML Form Adapter and HTTP Basic Adapter

Used in conjunction with Password Credential Validators (see *Manage password credential validators* on page 173). 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 428 and *HTTP Basic Adapter* on page 432).

OpenToken Adapter

Provides a generic interface for integrating with various applications, including Java- and .NET-based applications (see *OpenToken Adapter* on page 434).

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.

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 208).

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 209).

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 210).

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 211).

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 211).

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 212).

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 213).

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 30 in the "Supported Standards" chapter of the *Get started with PingFederate* guide). 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 HTTP Header Authentication Selector* on page 211).

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 web site (`www.pingidentity.com/en/products/downloads.html`). Additional authentication selectors may also be added to the download site periodically; contact your Ping Identity sales representative 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 539.

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 58)
- Access Token Management instances (see *OAuth access token management* on page 186)
- Password Credential Validators (see *Manage password credential validators* on page 173)
- Identity Store Provisioners (see *Configure Identity Store Provisioners* on page 313)

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 28 in the *Get started with PingFederate* guide.)

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

- *Account linking* on page 67
- *Account mapping* on page 67

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: An SP PingFederate uses a default, HyperSQL database to handle account linking. You can use your own data store instead, as needed. For more information, see *Configure an LDAP connection* on page 145.

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 448). 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 305). 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.

About 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 adapter 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 (see *Adapter contracts* on page 69).
- Set up connections to local data stores (see *Data stores* on page 70).
- 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 (see *SSO integration kits and adapters* on page 64 or *Token processors and generators* on page 58).
- Selectively mask attribute values recorded in transaction logs (see *Attribute masking* on page 71).

Attribute contracts

An attribute contract represents an agreement between an SP and an IdP about user attributes sent in a SAML assertion, either for browser-based SSO or WS-Trust STS. The contract is a list of case-sensitive attribute names. IdPs and SPs 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 71).

For an IdP, the attribute contract defines which attributes PingFederate sends in an assertion. While this contract is fixed for all users authenticating to the SP 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 485)

For an SP, the attribute contract defines the attributes PingFederate expects in a SAML assertion. 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 309 or *Manage token generators* on page 402). 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 69 below or *STS token contracts* on page 69).

The attribute contract must contain the attribute `SAML_SUBJECT`, 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 *Create an attribute contract* on page 259).

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 388). (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 59). Namespaces can be specified only for attributes of a WS-Trust IdP configuration using SAML 1.1 as the default token type (see *Define an attribute contract for IdP STS* on page 388).

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 64 and *Manage IdP adapters* on page 238).

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 309.)

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 331).

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 58 and *Manage token processors* on page 381).

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 402). 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 406).

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 local data stores to supply attributes for either the IdP's attribute contract, the SP's adapter contract, or STS token contracts (see sections above). Standard data stores may include any JDBC-accessible database or an LDAP v3-compliant directory server (see *Manage data stores* on page 143).

Alternatively, you can use the PingFederate Custom Source SDK to create your own driver for non-JDBC/LDAP data stores—including, for example, flat files or SOAP-connected databases (see the PingFederate *SDK developer's guide on page 539*).

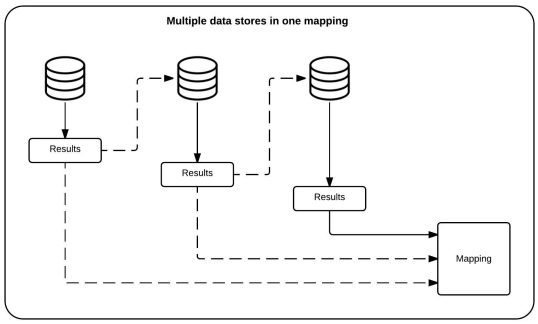
Data stores can be used across multiple connections.

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*.

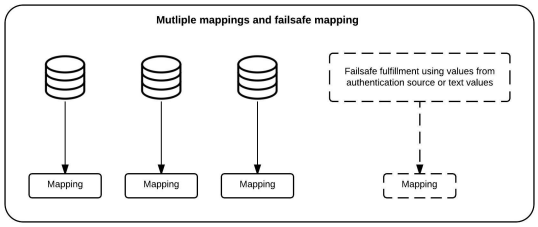
If a required attribute (such as `SAML_SUBJECT` in a SAML contact or `subject` in an authentication policy contract) cannot be fulfilled, the request fails.

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.



Attribute masking

At runtime PingFederate logs user attributes (see *PingFederate log files* on page 89). 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 143).
- Retrieval of attributes from an IdP adapter or token processor (see *Set pseudonym and masking options* on page 240 and *Set attribute masking* on page 385).
- SP-server processing of incoming attributes based on the SSO attribute contract, (see *Define an attribute contract* on page 330).

  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 67).
- SP-server processing of incoming attributes in response to an Attribute Request under XASP (see *Configure security policy for Attribute Query* on page 353).

  For information about XASP, see *Attribute Query and XASP* on page 42 in the “Supported Standards” chapter of the *Get started with PingFederate* guide.

  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 89).

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 *Specify issuance criteria (optional)* on page 275, as an example).

Issuance criteria

The token-authorization configuration consists of one or more rules that evaluate attribute values against selected conditions (see *Single and multi-value conditions* on page 72 below). You can choose from among several

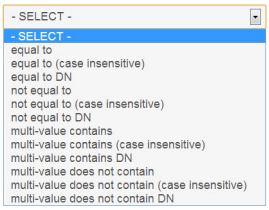
sources for the attributes, depending on the type of configuration that contains the token-authorization setup. The sources always consist of all of those available for attribute mapping, including data stores (when configured) and runtime information related to the context of an event. In addition, values of mapped attributes are available to provide access to any plain-text mappings or the runtime results of any expression mappings (see *Attribute mapping expressions* on page 485).

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 OGNL expressions using an Advanced Criteria feature. For more information, see *Attribute mapping expressions* on page 485 and *Sample OGNL expressions* on page 487.

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. For more information, see Error Result descriptions in sections of this document covering specific token-authorization configurations.

Single and multi-value conditions

Each token-authorization configuration provides a choice of conditions for evaluating attribute values:



Use one of the first six choices only for attributes consisting of a single value.

Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

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 (see *Security management* on page 160).

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 and decryption keys and certificates* on page 164).

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 (see *Configure certificate revocation* on page 170).

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 (see *Configure certificate revocation* on page 170). 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*.

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.

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 163).

  Important: The SSL/TLS server-client handshake involves negotiating cipher suites to be used for encryption/decryption on each side of a secured transaction. PingFederate supports only stronger cipher suites; to enhance security, weaker cipher suites are commented out of two configuration files located in `<pf_install>/server/default/data/config-store`:

  `com.pingidentity.crypto.SunJCEManager.xml`

  `com.pingidentity.crypto.NcipherJCEManager.xml`

  To ensure the most secure transactions, we recommend that administrators retain this cipher-suite configuration.

  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 web site* (www.oracle.com/technetwork/java/archive-139210.html).

- 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 and decryption keys and certificates* on page 164).

Verifying 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.

XML 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.

Using 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* in the *Get started with PingFederate* guide.)

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 306 or *Configure IdP Auto-Connect* on page 376.

Providing metadata

You enable Auto-Connect as part of Server Settings from the Main Menu (see *Choose roles and protocols* on page 136). 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 77 and *Configure metadata lifetime* on page 141).

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:

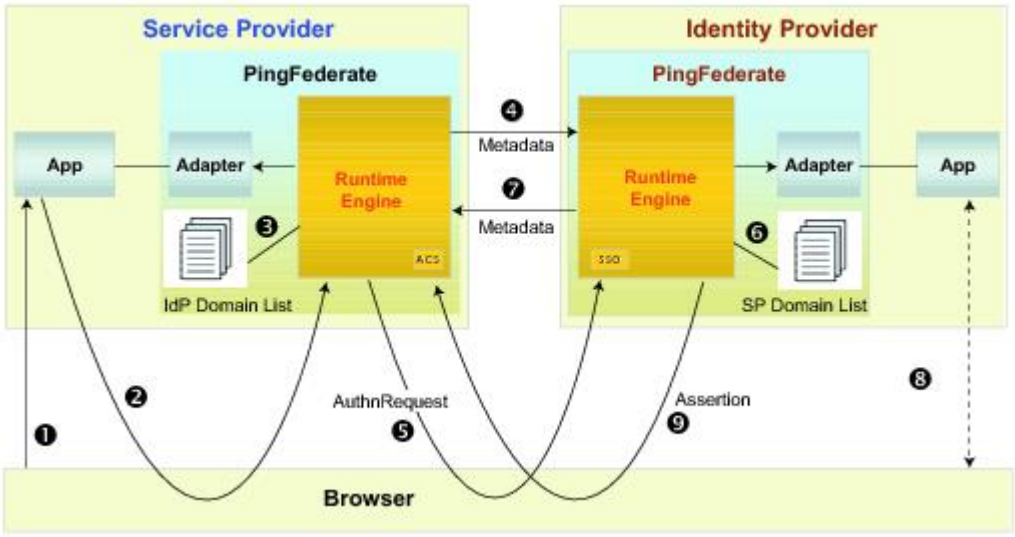


Figure 21: Auto-Connect Processing Flow

Processing Steps

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

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

`https://hostname.com:9031/sp/startSSO.ping/?Domain=mycompany.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.mycompany.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 77), 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:

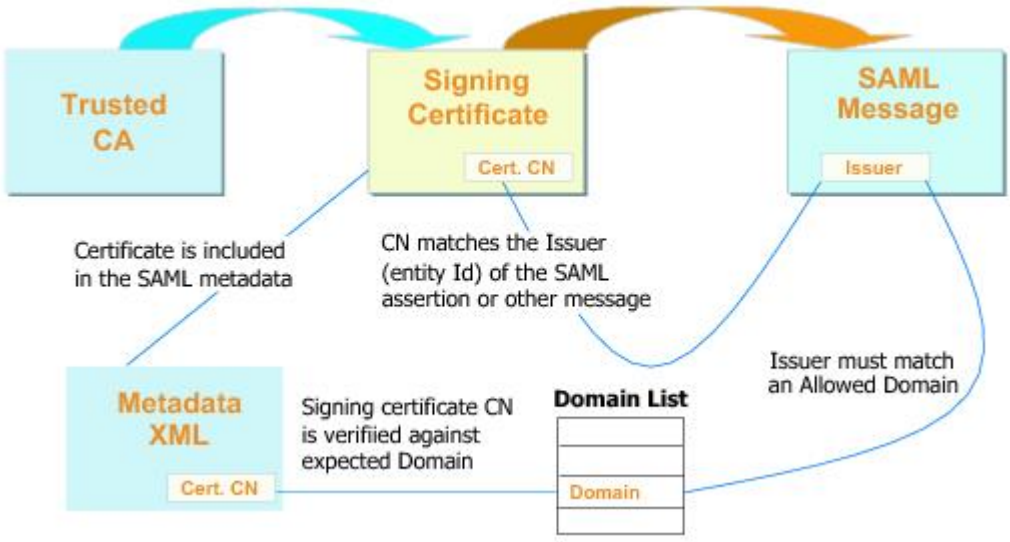


Figure 22: Auto-Connect Security Model

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.

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 78).

  For information about SCIM, see the web site *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 79).

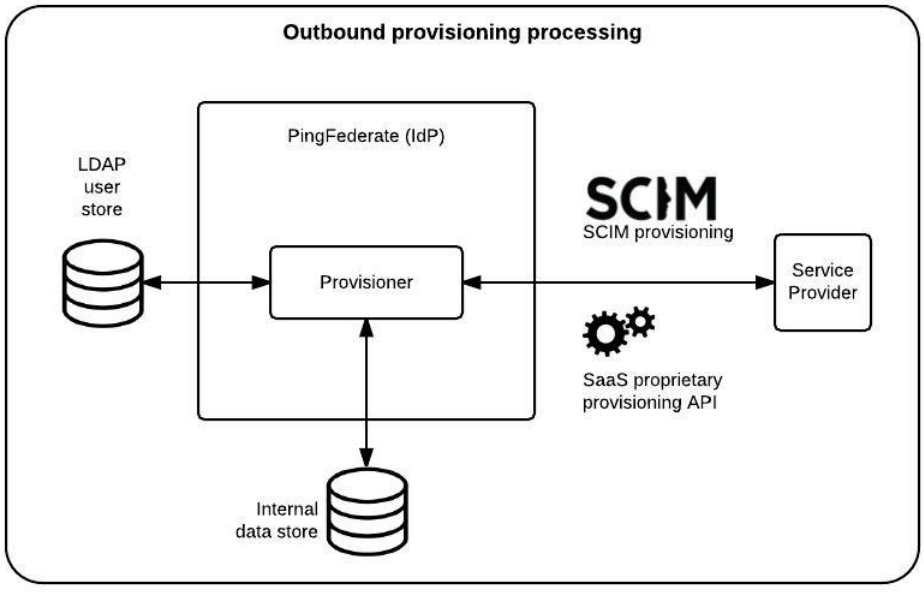
Outbound provisioning for IdPs

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.

Tip: Support for provisioning for SaaS applications, including quick-connection templates for partner SSO—SaaS connectors—is available separately from Ping Identity. Contact *sales@pingidentity.com* for more information.

Outbound provisioning also provides an automated means of account disabling or deprovisioning, which may be of key importance to system auditors.

When outbound provisioning is enabled, the PingFederate runtime engine 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:



As user-data sources, PingFederate provides built-in support for Microsoft Active Directory and Oracle Directory Server; templates are used to preconfigure many provisioning settings. Although these are the only data stores formally tested and supported, other LDAP data stores will likely work as well.

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

Provisioning for SPs

When PingFederate is enabled as an SP, two provisioning choices are available:

- *Inbound provisioning* on page 79 for SCIM requests coming either from inside or outside an organization
- *Just-in-time provisioning* on page 79 for creating and updating user accounts based on information contained in SAML assertions

Inbound provisioning

Inbound provisioning provides support for incoming SCIM messages containing requests to create, read, update, or delete/deactivate user and group records in Microsoft AD 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 type (see *Connection types* on page 57).

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 onboarding SaaS product—or from trusted partner IdPs.

For setup information, see *Configure inbound provisioning* on page 359. To integrate Inbound Provisioning with custom user stores, see *Configure Identity Store Provisioners* on page 313. For application-development information about using PingFederate endpoints for SCIM provisioning, see *SCIM inbound provisioning endpoints* on page 449.

Just-in-time provisioning

At an SP site, PingFederate can also 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*. This feature (formerly called Express Provisioning) 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 (see *Attribute Query and XASP* in the "Supported Standards" chapter of the *Get started with PingFederate* guide).

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 322. For configuration information, see *Configure just-in-time provisioning* on page 353.

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 *Using Auto-Connect* on page 75).

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* chapter of the *Get started with PingFederate* guide.)

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 72.)

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 *Trust models* on page 74).

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 515*).

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 82).

With PingFederate, you define a unique ID for each supported protocol (see *Specify federation information* on page 137). Optionally, you can also use a list of multiple *Virtual Server IDs* on a connection-by-connection basis (see *Multiple virtual server IDs* on page 81).

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 *Virtual host names* on page 111).

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 70).

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 64 and *Token processors and generators* on page 58.)

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 89.)

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 66).

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 68.)

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 *Using Auto-Connect* on page 75.

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 101 and *System-services endpoints* on page 452 for more information).

After providing the protocol endpoints information to Office 365, when Office 365 sends login requests to PingFederate, PingFederate picks the correct IdP adapter to authenticate the end users based on the virtual server ID in the requests.

For each successful login, PingFederate builds an assertion with issuer being set to the corresponding virtual server ID. When Office 365 receives the assertion, it creates the end user session with the right subdomain settings based on the issuer value in the assertion.

Connecting to a partner in multiple connections

In this use case, you connect to your partner in multiple connections. In each connection, you identify yourself and your partner differently.

For example, you as the SP provide separate environments for the end users based on their regions. Your IdP operates in two regions, Europe (EU) and North America (NA); their federation IDs are eu.idp.local and na.idp.local, respectively.

In the PingFederate SP server, you can create two IdP connections to federate identities for end users from both regions as follows:

	Partner's federation ID	Your virtual server ID
IdP connection #1	`eu.idp.local`	`idp-eu.sp.tld`
IdP connection #2	`na.idp.local`	`idp-na.sp.tld`

Based on the issuer (the partner's federation ID) and the audience values (your virtual server ID), PingFederate determines at runtime which IdP connection the assertion is intended for, validates as per the connection settings, and passes attribute values to the SP adapter to create the end-user session.

Working with multiple virtual server IDs

You can assign virtual server IDs either as an IdP during configuration of an SP connection (see *Identify the SP* on page 253) or as an SP configuring an IdP connection (see *Identify the IdP* on page 324) 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 442 as an IdP or *SP endpoints* on page 444 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 264, *Restrict a target session to certain virtual server IDs* on page 334 and *Restrict a target session to certain virtual server IDs* on page 334).

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 391 or *Restrict a token generator to certain virtual server IDs* on page 407).

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 *Specify issuance criteria (optional)* on page 275 as an IdP or *Identify issuance criteria (optional)* on page 341 as an SP.

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 *About attributes* on page 67).
- **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.

Federation hub

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 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 83
- *Bridging an IdP to multiple SPs* on page 84
- *Bridging multiple IdPs to an SP* on page 84
- *Bridging multiple IdPs to multiple SPs* on page 85

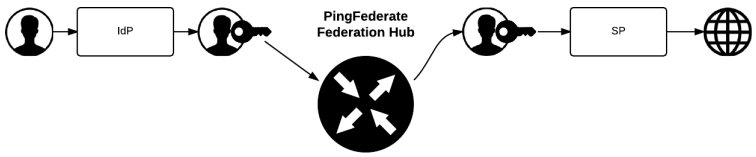
PingFederate also supports protocol translation among SAML 1.0, 1.1, 2.0 and WS-Federation. For SAML based connections, this also means it is possible to bridge between various bindings between identity providers and service providers.

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.

Note that the Single Logout (SLO) profiles are not supported for federation hub use cases.

Bridging an IdP to an SP

In this use case, PingFederate is bridging SSO 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.



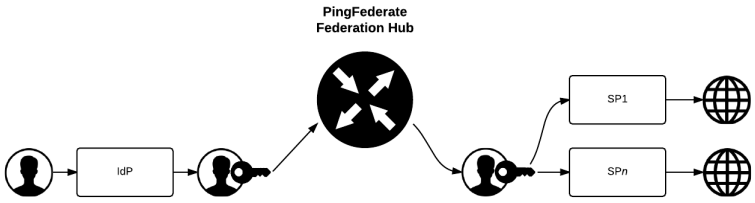
To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols (see *Choose roles and protocols* on page 136).
2. Create a contract to bridge the attributes between the identity provider and the service provider (see *Deploying PingFederate as a federation hub* on page 86).

3. Create an IdP connection between the identity provider and PingFederate (the federation hub as the SP). In the Target Session Mapping screen, add the authentication policy contract into the connection (see *Manage target session mappings* on page 331).
4. Create an SP connection between PingFederate (the federation hub as the IdP) and the service provider. In the Authentication Source Mapping screen, add the authentication policy contract into the connection (see *Manage authentication source mappings* on page 261).
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 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.

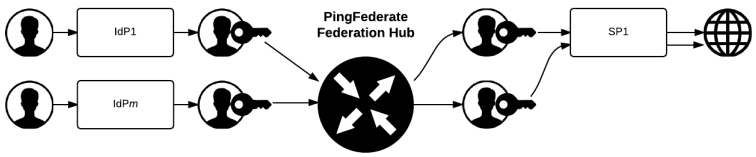


To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols (see *Choose roles and protocols* on page 136).
2. For each service provider, create a contract to the identity provider (see *Deploying PingFederate as a federation hub* on page 86). 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). In the Target Session Mapping screen, add the authentication policy contracts into the connection (see *Manage target session mappings* on page 331).
4. For each service provider, create an SP connection between PingFederate (the federation hub as the IdP) and the service provider. In the Authentication Source Mapping screen, add the corresponding authentication policy contract into the connection (see *Manage authentication source mappings* on page 261).
5. For each service provider supporting the SAML IdP-initiated SSO profile, map the expected Target Resource URLs to the corresponding SP connection (see *Configure target URL mapping* on page 311).
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 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 (see *Choose roles and protocols* on page 136).
2. Create a contract to bridge the attributes between the identity providers and the service provider (see *Deploying PingFederate as a federation hub* on page 86). 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). In the Target Session Mapping screen, add the corresponding authentication policy contract into the connection (see *Manage target session mappings* on page 331).
4. Configure an authentication selector to map each identity provider to the corresponding IdP connection (see *Manage authentication selectors* on page 208).
5. Create an SP connection between PingFederate (the federation hub as the IdP) and the service provider. In the Authentication Source Mapping screen, add the authentication policy contract into the connection (see *Manage authentication source mappings* on page 261).

   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.

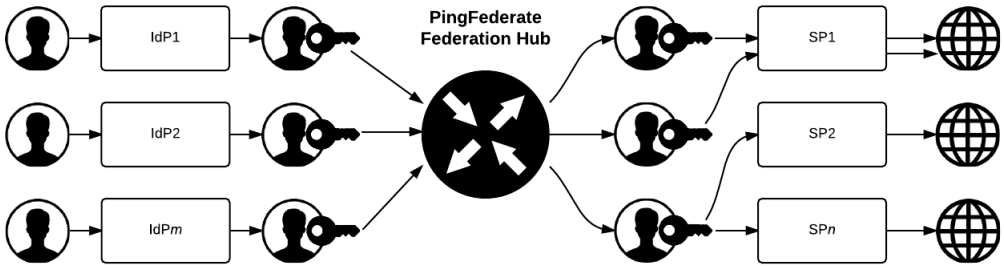
   For SAML 1.x assertions and WS-Federation security tokens, you can add an attribute to the Attribute Contract (see *Create an attribute contract* on page 259) and map `Authenticating Authority` into its value (see *Configure attribute contract fulfillment* on page 273).

   For information about `Authenticating Authority`, see section 2.7.2.2 Element <AuthnContext> in SAML 2.0 specification (*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 use Issuance Criteria in the IdP connection 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 use case is a combination of *Bridging an IdP to multiple SPs* on page 84 and *Bridging multiple IdPs to an SP* on page 84.



To address this use case:

1. Enable both the IdP and the SP roles with the applicable protocols (see *Choose roles and protocols* on page 136).
2. Create multiple contracts to bridge the attributes between the identity providers and the service providers (see *Deploying PingFederate as a federation hub* on page 86).
3. For each identity provider, create an IdP connection between the identity provider and PingFederate (the federation hub as the SP). In the Target Session Mapping screen, add the applicable authentication policy contract(s) into the connection (see *Manage target session mappings* on page 331).
4. Configure an authentication selector to map each identity provider to the corresponding IdP connection (see *Manage authentication selectors* on page 208).
5. For each service provider, create an SP connection between PingFederate (the federation hub as the IdP) and the service provider. In the Authentication Source Mapping screen, add the corresponding authentication policy contract into the connection (see *Manage authentication source mappings* on page 261).

   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 to the Attribute Contract (see *Create an attribute contract* on page 259) and map `Authenticating Authority` into its value (see *Configure attribute contract fulfillment* on page 273).

   For information about `Authenticating Authority`, see section 2.7.2.2 Element <AuthnContext> in SAML 2.0 specification (*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 use Issuance Criteria in the IdP connection to protect against user impersonation between IdPs.
6. For each service provider supporting the SAML IdP-initiated SSO profile, map the expected Target Resource URLs to the corresponding SP connection (see *Configure target URL mapping* on page 311).
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).

Deploying PingFederate as a federation hub

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.

For information about managing authentication policy contracts, see *Manage policy contracts* on page 235.

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 (see *Identify the SP* on page 253), 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 (see *Specify federation information* on page 137).

If the SP connection uses multiple virtual server IDs (see *Connecting to a partner in one connection* on page 81), for SP-initiated SSO, if the service provider sends AuthnRequest messages to the virtual server ID specific endpoint (see *step 3* under **To export connection metadata** in *Export metadata to an XML file* on page 101), 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.

System administration

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

- *Start and stop PingFederate*
- *PingFederate log files* on page 89
- *Export metadata to an XML file* on page 101
- *Sign XML files* on page 103
- *Manage configuration archives* on page 103
- *Account management* on page 105
- *Alternative console authentication* on page 108
- *Manage email configuration* on page 110
- *Virtual host names* on page 111
- *PingFederate properties* on page 112
- *Manage PingFederate license* on page 88

- *Automating configuration migration* on page 114
- *Outbound provisioning CLI* on page 117

Note: The information in this chapter is presented from the viewpoint of an administrative user with "Admin" permissions (see *Account management* on page 105).

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 installation. Depending on your licensing agreement, your PingFederate license may have an expiration date. Furthermore, you may need a new license key for a PingFederate upgrade (other than maintenance releases).

If your license key is going to expire (or has expired recently), you can replace the current license key with a new one without interrupting services by importing the replacement license file via the administrative console. You may also configure PingFederate to send administrators email warnings regarding the license status.

To review the current license key:

1. Start PingFederate and sign-on to the administrative console.
2. Click the account 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.

Request a new license key

1. Go to the Ping Identity *License Key Request Form* (`www.pingidentity.com/support/licensing`).
2. Sign on, provide the required information, and 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**.

   Once the license file is verified for use with the current instance of PingFederate, it is copied to the `<pf_install>/pingfederate/server/default/conf` directory. The file name is `pingfederate.lic`. This is referred as *the primary license file*.

In a clustered PingFederate environment, the running engine nodes consume the new license and save the information to the `<pf_install>/pingfederate/server/default/data/.pingfederate.lic` license file. This is referred to as *the secondary license file*. Note that the secondary license file is used only when the primary license file cannot be found.

Tip: You are not required to restart PingFederate on any nodes.

For any engine node that is stopped when a new license is imported through the administrative console, the new license is utilized when the engine node restarts. The new license information is saved to the secondary license file.

Install a replacement license key

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**.

   Once the license file is verified for use with the current instance of PingFederate, it is copied to the `<pf_install>/pingfederate/server/default/conf` directory. The file name is `pingfederate.lic`. This is referred as *the primary license file*.

The previous `pingfederate.lic` license file is renamed with a timestamp in the same `conf` directory.

If the new license does not include support for features covered by your existing license, or if there is some other potential problem with the license, you are advised and prompted on whether to continue.

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 reverts to the previous license.

In a clustered PingFederate environment, the running engine nodes consume the new license and save the information to the `<pf_install>/pingfederate/server/default/data/.pingfederate.lic` license file. This is referred to as *the secondary license file*. Note that the secondary license file is used only when the primary license file cannot be found.

Tip: You are not required to restart PingFederate on any nodes.

For any engine node that is stopped when a new license is imported through the administrative console, the new license is utilized when the engine node restarts. The new license information is saved to the secondary license file.

Configure notification for licensing events

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 you back to the **Runtime Notifications** screen. Click **Save**.

PingFederate log files

PingFederate® generates these logs that document server events:

- `admin.log` — Records actions performed by administrative-console users (see *Administrator audit logging* on page 91).
- `admin-event-detail.log` — Records detailed information about each applicable administrative-console event performed by administrative-console users if detailed event logging is enabled (see *Detailed event logging* on page 91).
- `admin-api.log` — Records actions performed by administrative-api users (see *Administrative API audit log* on page 92).
- `transaction.log` — Records individual identity-federation runtime transactions at specified levels of detail.

  The level of detail can be set globally or on a connection-by-connection basis (see *Runtime transaction logging* on page 93).
- `audit.log` — Records a selected, configurable subset of transaction log information plus additional details, intended for security-audit and regulatory compliance purposes (see *Security audit logging* on page 93).
- `provisioner-audit.log` — Records Outbound Provisioning events, intended for security-audit purpose (see *Outbound provisioning audit logging* on page 95).
- `server.log` — Records all PingFederate runtime and administrative server operational activity (see *Server logging* on page 95). PingFederate provides a utility for filtering server log entries (see *Using the server log filter* on page 96).
- `provisioner.log` — Records only provisioning activity.

  (For more information, see *Outbound provisioning for IdPs* on page 78).
- `init.log` — Records only Jetty messages generated prior to PingFederate start up.

Tip: PingFederate logs user attributes, when they are present, in the server log, the transaction log, and the audit log (if configured). When privacy is required for sensitive user attributes, you can configure PingFederate to obfuscate (mask) their values in these logs (see *Attribute masking* on page 71).

The logs are stored by default in the <pf_install>/pingfederate/log directory. You can change the location to any network directory by using the runtime parameter pf.log.dir (see *PingFederate properties* on page 112).

The audit log, server log, provisioner log, and provisioner audit log may be output to alternate formats, including database tables (see *Writing logs to other formats* on page 97).

The PingFederate-generated logs are controlled through the log4j2.xml file located in <pf_install>/pingfederate/server/default/conf directory. See comments in the file for more information.

Note: By default, initial server requests are always recorded in the server log with a tracking ID number, which is then used to identify subsequent, related transactions. This ID can be useful for debugging and support purposes to aggregate and analyze log entries tied to the same original request. The ID may also be added to the configuration for transaction.log or audit.log, or it may be removed from the server.log <layout> element in the log4j2.xml file, as needed for performance considerations.

Refer to the *log4j2 open-source project* for more information about logging levels and other configuration parameters (logging.apache.org/log4j/2.x/manual/index.html).

By default, PingFederate installs with a highly verbose level of logging. However, verbose logging may have a performance impact and clutter the log files. You may choose to lower the level, but we recommend that you not set it below WARN. For the transaction log and the audit log, note that any setting below INFO turns logging off.

Important: 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 transaction.log, audit.log, audit-event-detail.log (if enabled), and provisioner-audit.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.

server.log is rolled over when it reaches 10 MB. The system keeps five old log files before overwriting the oldest. (The file size and the number of files to be kept can be changed in the log4j2.xml file.)

Updating the log level for an existing logger does not require a restart. The adjustments are activated within half a minute. Other changes require a restart of PingFederate; for examples:

- 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

Note: log4j2.xml is individually managed per PingFederate server. This flexibility allows multiple PingFederate nodes in a clustered environment to write different level of messages to different destinations.

The following sections provide more information about the primary logs:

- *Administrator audit logging* on page 91
- *Administrative API audit log* on page 92
- *Runtime transaction logging* on page 93
- *Security audit logging* on page 93
- *Outbound provisioning audit logging* on page 95
- *Server logging* on page 95
- *Writing logs to other formats* on page 97

HTTP requests to the runtime engine and administrative console are logged to pf.log.dir/<date>.request.log and pf.log.dir/<date>.request2.log, respectively, by the Pingfederate web container. Properties controlling request logging are contained in the Web-container configuration files located in the <pf_install>/pingfederate/etc directory, namely jetty-runtime.xml (for <date>.request.log) and jetty-admin.xml (for <date>.request2.log).

Administrator audit logging

PingFederate records actions performed by server administrators. This information is recorded in the `admin.log` file. While the events themselves are not configurable, `log4j2.xml` configuration settings 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

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 `admin.log`, PingFederate logs detailed information about each event to `admin-event-detail.log`. 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 `pf.log.eventdetail` to `true` in `<pf_install>/pingfederate/bin/run.properties` (see *PingFederate properties*.)

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 below:

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 (see *PingFederate administrative API*). This information is recorded in the `admin-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.

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 `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.

For information about each service, see:

- *OAuth Client Management Service*
- *OAuth Access Grant Management Service*
- *Session revocation API*

Runtime transaction logging

PingFederate provides for flexible, scalable logging of all federated-identity transactions (inbound and outbound XML messaging). Transaction logging can be configured to any of four modes on a connection-by-connection basis (see “General Information” in either the IdP or SP “SSO Configuration” chapters).

You also have the option of overriding transaction logging for all connections (see *Via the Manage Connections screen* on page 246 for SP connections or *From the Manage Connections screen* on page 319 for IdP connections). You may want to use this override for troubleshooting or as a one-step means of raising or lowering all connection logging modes to the same level.

Transaction logging modes

The table below describes the four transaction logging modes:

Table 2: Transaction Logging Modes

Mode	Description
None	No transaction logging.
Standard	(Default) Logs summary information for each transaction message, including: • Timestamp • Hostname:Port • Log Mode • Connection ID • SAML Status Code (for SAML responses only) • Context • Message Type • SAML ID (for SAML messages only) • Endpoint (for outbound messages only) • Target URL (if SSO transaction)
Enhanced	Includes everything logged at the Standard level plus: • SAML_SUBJECT* • Binding • Relay State (if available) • Signature Policy • Signature Status • HTTP Request Parameters (outbound messages only) * Only when available in a SAML assertion, a single-logout request, a 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 of the above fields is separated by a vertical pipe (|) for easier parsing.

Security audit logging

The PingFederate records a subset of transaction log information with additional details at runtime, intended to facilitate security auditing and regulatory compliance. The following table describes the default elements that PingFederate writes to the audit log (in the order that they are listed):

Element	Description
%d	Transaction time.

Element	Description
event	The type of transaction; for example, `SSO`.
subject	The subject of the transaction.
ip	Incoming IP address.
app	The target SP application (when available).
connectionid	The connection identifier associated with the transaction.
protocol	The associated identity protocol; for examples, `SAML20` or `OAuth20`.
host	PingFederate host name or IP address.
role	The role of PingFederate played for the transaction.
status	Transaction success or failure.
adapterid	The ID of an adapter instance.
description	Description of an authentication failure (when information is available from an IdP adapter).
responsetime	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.

PingFederate records this information in the `audit.log` file located in the `<pf_install>/pingfederate/log` directory. Each element is separated by a vertical pipe (`|`). Elements are configurable by editing the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.

The following table describes other elements (in alphabetical order) that can be added to the audit log:

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	User attributes received (for an SP log) or sent (for an IdP log).
authenticationsourceid	An array of one or more IdP adapters, IdP connections, or both, invoked in an authentication or logout flow.
granttype	OAuth grant type.
initiator	(SAML 2.0 only) The federation role that initiated the SSO or SLO: `SP` or `IDP`.
inmessagetype	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 examples, 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 examples, a SAML Response or the information pertaining to a response for an OAuth request.
pfversion	The PingFederate version.
requestid	The ID of a request.
responseid	The ID of a response.

Element	Description
requeststarttime	The start time of the request in milliseconds since midnight, January 1, 1970 UTC.
stspluginid	For WS-Trust STS transactions, the ID for the token-processor or token-generator instance.
targetsessionid	An array of one or more SP adapters, SP connections, or both, invoked in an authentication or logout flow.
trackingid	The tracking ID used for debugging purposes in the server log.
validatorid	The ID of the Password Credential Validator instance used for OAuth resource-owner grant transactions.
virtualserverid	The virtual server ID of a request (if applicable).

The `audit.log` file is rolled over at midnight daily. As needed, administrators may choose a different formats, including databases. For more information, see *PingFederate log files* and *Writing logs to other formats*.

Tip: The audit log records SSO, SLO, OAuth AS, WS-Trust STS, and Inbound SCIM transactions. Outbound Provisioning transactions are not included; they are logged to the provisioner audit log (see *Outbound provisioning audit logging*).

Outbound provisioning audit logging

The PingFederate `provisioner-audit.log` records Outbound Provisioning transactions, intended to facilitate security auditing. Elements of this log are described in the following table and configurable in the `log4j2.xml` file located in `pingfederate/server/default/conf`.

Important: Monitoring Outbound Provisioning transactions using JMX has been deprecated. By default, PingFederate logs outbound provisioning events to `provisioner-audit.log` (see *Runtime monitoring using JMX* on page 131).

Elements from the provisioner audit log may be output to different formats, including databases. For information, see *Writing logs to other formats* on page 97.

Table 3: Provisioner Audit Log Configuration

Item	Description
%d	Transaction time.
cycle_id	The unique ID for each provisioning cycle.
channel_id	The unique ID for 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

The server log records all PingFederate runtime and administrative events, including status and error messages that can be used for troubleshooting. By default, the information is also sent to the terminal or command window running the PingFederate server.

Note: Alternatively, you can output the server log to a database (see *Writing logs to other formats* on page 97).

To facilitate troubleshooting, administrators can use a filter utility to aggregate related events (see *Using the server log filter* on page 96).

Elements of this log are described in the following table.

Table 4: Server Log Components

Item	Description
%d	Event date and time.
%X{trackingid}	The tracking ID for runtime events, used for debugging purposes (see the last *Note* in the section *PingFederate log files* on page 89).
%p	Logging level.
%c	The Java class issuing the status or error message, when applicable.
%X{connectionid}	The partner ID associated with a runtime event.
%X{subject}	The SAML subject of the transaction, when applicable.
%m	Status or error message.

Using the server log filter

PingFederate provides a utility that administrators can use to filter server logs. The tool, `logfilter.bat|sh`, is located in the `<pf_install>/pingfederate/bin` directory.

By default the utility sorts through all the server logs in the log directory, or you can move or copy one or more 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.

Table 5: Server Log Filter Command Options

Command Option	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_xref_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_xref_id> may be the value of any of the following transaction tags in the target server log(s): • Artifact • Session Index • Assertion ID

Table 6: Server Log Filter Parameters*

Parameter*	Description
-logsdir <log-files-directory>	Full or relative path to source directory for the log(s). Default: all `server.log` files in `pf.log.dir`, i.e., `<pf_install>/pingfederate/log` (see *PingFederate properties* on page 112).
-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.

*Optional for all commands

The log filter creates its own log file, `logfilter.log`, located in the log directory. You can control settings for this log, as needed, in the file `logfilter.log4j2.xml`, located in the `<pf_install>/pingfederate/bin` directory.

Writing logs to other formats

PingFederate provides the option of writing the audit, server, provisioner, and provisioner audit logs to commonly used databases (with failover to file logging).

For the audit log and the provisioner audit log, you may choose instead to write the information to the Common Event Format (see *Write audit or provisioner audit logs to CEF* on page 99).

Finally, you may also configure PingFederate to write the audit log to a differently formatted log file that can be used by Splunk (see *Write audit logs for Splunk* on page 99).

Write logs to databases

Database logging replaces file logging. For the server log, database logging also replaces logging to the terminal or command window running PingFederate.

Failover file logging is provided in the event that database logging fails for any reason. By default, PingFederate retries database logging every minute. Messages written to log files during failover periods are not copied over to the database server.

Scripts to create database tables for each of the four databases are provided for the audit log, server log, provisioner log, and provisioner audit log. The scripts are located in the `<pf_install>/pingfederate/server/default/conf/log4j/sql-scripts` directory.

Important: Ensure that your JDBC 4.1 (or higher) database-driver JAR file is installed in the `<pf_install>/pingfederate/server/default/lib` directory. You must restart the server after installing the driver.

You enable database logging for the audit log, the server log, the provisioner log, and the provisioner audit log in the `log4j2.xml` file in the `<pf_install>/pingfederate/server/default/conf` directory.

1. In `log4j2.xml`, uncomment one of the preset `JDBC` log-appender configurations listed below (or one from each list to configure all logs):

   For the server log:

   - `ServerLogToOracleDB` (for Oracle)
   - `ServerLogToSQLServerDB` (for Microsoft SQL Server)
   - `ServerLogToMySQLDB` (for MySQL)

   For the provisioner log:

   - `ProvisionerLogToOracleDB` (for Oracle)
   - `ProvisionerLogToSQLServerDB` (for Microsoft SQL Server)

- `ProvisionerLogToMySQLDB` (for MySQL)

For the audit log:

- `SecurityAuditToOracleDB` (for Oracle)
- `SecurityAuditToSQLServerDB` (for Microsoft SQL Server)
- `SecurityAuditToMySQLDB` (for MySQL)

For the provisioner audit log:

- `OutboundProvisionerEventToOracleDB` (for Oracle)
- `OutboundProvisionerEventToSQLServerDB` (for Microsoft SQL Server)
- `OutboundProvisionerEventToMySQLDB` (for MySQL)

Note: Each appender is followed by a related appender (`RollingFile`) that creates a running `*failover.log` file in the log directory. The failover appender (`PingFailover`) must also be enabled (uncommented).

2. Replace placeholder parameter values in `log4j2.db.properties` in the same directory for the applicable appenders.

   The parameter values provide access to the database. We recommend that they be tested and validated 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.

   Note: See the NOTES in `log4j2.xml` above the appender for more details.

   Tip: You can obfuscate the password used to access the database by running either `obfuscate.sh` or `obfuscate.bat`, located in `<pf_install>/pingfederate/bin`. Use the actual password as an argument and copy the entire result into the value for the password parameter in `log4j2.xml`.

3. Uncomment the appender reference in the associated logger elements, as described in the appender `NOTES`:

   For the server log:

   Uncomment the appender reference located under `Set up the Root Logger` near the end of the `log4j2.xml` configuration file.

   For the provisioner log:

   Uncomment the appender reference located under `Limit categories` near the end of the `log4j2.xml` configuration file.

   For the audit log:

   Uncomment the appender in one or more of the following loggers located under `Limit categories` in the `log4j2.xml` configuration file:

   - `org.sourceid.websso.profiles.idp.SpAuditLogger` (Browser SSO IdP and Adapter-to-Adapter)
   - `org.sourceid.websso.profiles.sp.IdpAuditLogger` (Browser SSO SP 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)

   For the provisioner audit log:

   Uncomment the appender reference located under `Limit categories` near the end of the `log4j2.xml` configuration file.

   Note: As indicated in the IMPORTANT comment for the loggers, you must remove the existing appender references.

4. Optional: For the audit log and the provisioner audit log, you can configure elements for database logging in the `ConversionPattern` appender parameter, as needed. For more information, see *Security audit logging* on page 93 and *Outbound provisioning audit logging* on page 95, respectively.

Write audit or provisioner audit logs to CEF

The Common Event Format (CEF) is an open logging standard. PingFederate provides an option of writing elements from audit log or provisioner audit log (or both) at runtime to a syslog receiver for parsing and analysis via HP ArcSight tools. Alternatively, you can write CEF to a flat file; however, using syslog, when available, is recommended.

You can enable this capability in the `log4j2.xml` file (in the `pingfederate/server/default/conf` directory).

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.

Configure CEF logging for the audit log

1. In `log4j2.xml`, uncomment one of the preset log-appender configurations:
   - `SecurityAuditToCEFSyslog`
   - `SecurityAuditToCEFFile`
2. Replace the placeholder parameter value for the syslog host (if you are configuring the syslog appender).
3. Uncomment the chosen appender in one or more of the following loggers located under `Limit categories` in the `log4j2.xml` configuration file:
   - `org.sourceid.websso.profiles.idp.SpAuditLogger` (Browser SSO IdP and Adapter-to-Adapter)
   - `org.sourceid.websso.profiles.sp.IdpAuditLogger` (Browser SSO SP 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)

Configure CEF logging for the provisioner audit log

1. In `log4j2.xml`, uncomment one of the preset log-appender configurations:
   - `OutboundProvisionerEventToCEFSyslog`
   - `OutboundProvisionerEventToCEFFile`
2. Replace the placeholder parameter value for the syslog host (if you are configuring the syslog appender).
3. Uncomment the chosen appender for the `ProvisionerAuditLogger` logger located under `Limit categories` in the `log4j2.xml` configuration file.

   Note: As indicated in the IMPORTANT comment for the loggers, you must remove the existing appender reference(s) for syslog implementations.

Write audit logs 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

  Note: OAuth AS and WS-Trust STS transactions are not currently supported for Splunk.
- 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 the Splunk Apps web site "*splunkbase*" (`splunkbase.splunk.com`)—search for `PingFederate`.

1. Download and install the application in your Splunk environment from (see the **Note** above).
2. In your PingFederate installation, open the file `log4j2.xml` in the directory:

   `<pf_install>/pingfederate/server/default/conf`
3. In the `log4j2.xml` file, depending on role of your PingFederate server, search for:
   - `Logger name="org.sourceid.websso.profiles.idp.SpAuditLogger` (Browser SSO IdP and Adapter-to-Adapter)
   - `Logger name="org.sourceid.websso.profiles.sp.IdpAuditLogger` (Browser SSO SP and Adapter-to-Adapter)
   - `Logger name="org.sourceid.websso.profiles.idp.AsAuditLogger` (OAuth Authorization Server)
   - `Logger name="org.sourceid.wstrust.log.STSAuditLogger` (WS-Trust STS, IdP and/or SP)
4. Replace the `ref` attribute value in the `<appender-ref>` element for the applicable logger(s) with the value for Splunk indicated in the associated comment.

   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="SecurityAuditToSQLServerDB-FAILOVER"/>
        <appender-ref ref="SecurityAuditToOracleDB-FAILOVER"/>
        <appender-ref ref="SecurityAudit2Splunk"/>
    -->
</Logger>
```

   To log IdP audit log messages to `splunk-audit.log`, update the attribute value of `ref` from `SecurityAudit2File` to `SecurityAudit2Splunk`:

```
<Logger name="org.sourceid.websso.profiles.idp.IdpAuditLogger"
    level="INFO" additivity="false" includeLocation="false">
    <appender-ref ref="SecurityAudit2Splunk" />
    <!--
        ...
    -->
</Logger>
```

   Note: For auditing of adapter-to-adapter events, you must enable both the IdP and SP loggers.
5. Save the `log4j2.xml` file.
6. Download and install the Splunk Universal Forwarder on the machine running PingFederate.

   Configure the Splunk Universal Forwarder to monitor the `splunk-audit.log` in the PingFederate directory `<pf_install>/pingfederate/log`.

For detailed installation and configuration instructions, consult the Splunk documentation: *Set up forwarding and receiving* (`docs.splunk.com/Documentation/Splunk/latest/Forwarding/Setupforwardingandreceiving`).

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** workflow 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** workflow 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** workflow 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. A signature applied 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.

When you configure a partner connection, you can also verify and import signed metadata files. For information:

- As an SP configuring an IdP connection, see *Import IdP metadata* on page 323.
- As an IdP configuring an SP connection, see *Import SP metadata* on page 251.

XML file signing is available from the **Main Menu** > **Server Configuration** > **XML File Signatures** screen.

1. On the **Select XML File** screen, locate and open the file.
2. On the **Digital Signature Settings** screen, choose the certificate containing your signing key from the drop-down 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 the "**... <KeyValue> ...**" check boxes.
3. Select a signing algorithm corresponding to the selected certificate. Choices include RSA-SHA1, SHA256, SHA384, and SHA512; ECDSA-SHA256, SHA384 and SHA512; as well as DSA SHA1.
4. On the **Export & Summary** screen, click the **Export** button to save the signed file.

   Important: To finish the download, be sure to click the **Export** button at the bottom left of the screen.

Replicate configuration

From the **Main Menu** > **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 529 in the PingFederate *Server clustering guide on page 515*.

Manage configuration archives

An administrator can export the current administrative-console configuration to a ZIP file and import an existing archive for immediate deployment into a running PingFederate server in 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 (see *Account management* on page 105).

A time-stamped configuration archive is created as a backup automatically every time an administrator logs 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. Configuration archives can be used as backups for the current PingFederate installation.

Caution: As the backup file contains your complete PingFederate configuration, ensure the file is protected with appropriate security controls in place.

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.

The archive utility is intended for administrative-console configuration data only. It does not include error-page or other end-user HTML templates (see *Customizable user-facing screens* on page 120), nor any files under the `<pf_install>/pingfederate/server/default/conf` directory. If any changes have been made to

the default templates or configuration files, such files must be copied over to new installations or other instances of PingFederate (assuming the changes are applicable).

The archive also does not capture license files, adapter JAR files, database drivers, or any other plug-ins; these also must be copied into any new instance of PingFederate. The import utility checks for and reports on any missing components (see *Import an archive* on page 104).

Tip: PingFederate includes a separate command-line utility that provides a finer level of control over configuration migration, as well as providing for scripting of routine configuration-management tasks (see *Automating configuration migration* on page 114).

Create an archive

1. On the **Select Import/Export** screen, select the Export option and click **Next**.
2. On the **Export** screen, click **Export**, save the download to your file system, and click **Done**.

Import and deploy an archive

- On the **Select Import/Export** screen, click the Import option and click **Next**.

  See *Import an archive* on page 104 for more information.

  Note: Alternatively, you can deploy an archive manually.

  Caution: Deploying a configuration archive, either manually or by using the administrative console, always overwrites all existing configuration data.

Deploy an archive manually

1. Copy a previously stored archive into this directory:

   `<pf_install>/pingfederate/server/default/data/drop-in-deployer`
2. Rename the copied file to `data.zip`.

   Caution: Deploying a configuration archive, either manually or by using the administrative console, always overwrites all existing configuration data.

   When the PingFederate server is running, the file is again renamed with a timestamp after a moment and the data automatically deploys, replacing the current UI configuration. Restarting the server is not required.
3. Consult the PingFederate server start-up window, or the server log file, for any messages concerning missing plug-in components or other errors.

   Note: A data archive imported via the `drop-in-deployer` directory is deployed by default regardless of errors, unlike the default behavior using the **Select Import/Export** screen.

Export an archive

1. In the **Export** screen, click **Export** and save the archive on your file system.
2. Click **Done**.

Caution: As the backup file contains your complete PingFederate configuration, ensure the file is protected with appropriate security controls in place.

Import an archive

When you initiate deployment of a configuration archive using the **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.

If there are missing components or license inconsistencies, the import is halted by default to allow you to install the necessary components or license. However, you 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 PingFederate server restart.

Caution: Deploying a configuration archive, either manually or by using the administrative console, always overwrites all existing configuration data.

1. On the **Import** screen, click **Browse** to locate the required ZIP file.
2. Optional: Select **Force Import** check box to deploy the archive regardless of whether dependencies are detected.

   Important: If you make this selection, consult the server start-up window or the server log for any errors.
3. Click **Import**.
4. Read the on-screen caution statement and indicate whether you want to continue.

   When you click **Yes**, a message is displayed indicating the deployment result.
5. Click **Done**.

Account management

The PingFederate administrative console supports four authentication schemes:

- Native authentication
- LDAP authentication
- RADIUS authentication
- Certificate-based authentication

For native authentication, the administrative console offers two system administration styles:

- **Single-User Administration** restricts access to only one account with full privileges.
- **Multi-User Administration** allows the creation of multiple console accounts with different roles. This is the default style.

Tip: You can update administration style in the **Server Configuration** > **Server Settings** > **System Administration** screen.

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.

Task	Steps
	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 retain your configuration.

Enable notification for password changes

User administrators may enable email notification to notify users about password changes.

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

1. Go to the **Server Configuration** > **Account Management** screen.
2. Select the **Notify User of Password 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 retain your configuration.

When email notification is enabled, PingFederate sends emails about password changes to the users automatically via the configured email server.

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 PingFederate, 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 logged 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 retain 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`.
SSL SMTP Port	The secure SMTP port on your email server. This field is not active unless the **Use SSL** check box is selected. The default value is `465`.
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`.
Use SSL	If selected, requires the use of the SMTP secure channel.
Use TLS	If selected, requires the use of the secure transport layer.
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 the **Use SSL** or **Use TLS** check box is selected.
Enable SMTP Debugging Messages	Turns on detailed error messages for the PingFederate server log to help troubleshoot any problems.
Username	Authorized email username.
Password	User password.
Confirm Password	
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.

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 in 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 for a federation deployment, normally in the *same* domain (see *Federation Server Identification*). Depending on your needs, however, you can configure virtual server IDs and virtual hosts in the same installation of PingFederate.

PingFederate properties

PingFederate's default administrative-console and runtime behavior is controlled in part by configuration properties contained in the file `run.properties`, located in the `<pf_install>/pingfederate/bin` directory. The table below describes the properties.

Tip: Refer to the file itself for default settings not specified here, including various cookie-encoding options.

Note: Properties related to server clustering and provisioning failover are described in the PingFederate *Server clustering guide* *on page 515*.

You can change these settings as needed. Restart the PingFederate server for changes to take effect.

Important: If PingFederate is deployed in a cluster, changes to default settings for runtime-server properties must be applied to other server nodes manually. The settings in `run.properties` are not replicated to other server nodes in the cluster.

Table 7: PingFederate Configuration Properties

Property	Description
pf.admin.https.port	Defines the port on which the PingFederate administrative console runs. Default 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 (see *Detailed event logging* on page 91).
pf.console.login.mode	Indicates whether more than one Admin user may access the administrative console at one time (see *Set administration options* on page 130). Values: `Single | Multiple`. Default is `Multiple`.
pf.console.authentication	Indicates whether administrators log on to PingFederate using credentials managed internally, by PingFederate, or externally (see *Alternative console authentication* on page 108).
pf.admin.api.authentication	Defines the authentication method of the PingFederate administrative API (see *Configure access to the administrative API* on page 481).
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.

Property	Description
pf.https.port	Defines the port on which PingFederate listens for encrypted HTTPS (SSL/TLS) traffic. Default 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 `WantClientAuth` or `NeedClientAuth` set to `true`—see "**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 (see *Configure runtime reporting* on page 131). Use for deployments where multiple network interfaces are installed on the machine running PingFederate.
pf.engine.prefer_ipv4	Defines the protocol to be used by PingFederate. `True` (the default) enables use of `ipv_4` only. `False` enables use of both `ipv_4` and `ipv_6`.
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 (see *Application endpoints* on page 442). **Note:** If this property is changed, the path must also be added to the Base URL for your server (see *Specify federation information* on page 137).
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 (see *Thales nShield Connect HSM* in the *Get started with PingFederate* guide).
pf.provisioner.mode	Enables or disables (the default) Outbound Provisioning (see *Outbound provisioning for IdPs* on page 78). Also used to enable provisioning failover (see the PingFederate *Server clustering guide on page 515*).
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 128).

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

`WantClientAuth` and `NeedClientAuth` flags, which indicate that a client certificate is either requested or required, respectively, for mutual SSL/TLS. (For the preconfigured SSL secondary port, `WantClientAuth` is set to `true` by default; `NeedClientAuth` is set to `false`.)

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 `<pf_install>/pingfederate/bin`, 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 Service* on page 466).

Important: The Connection Management Service must be activated for both the source and target servers before the configcopy tool can be used (see *Authentication* on page 172).

Caution: For security reasons, the 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.

To copy the key from the source to the target servers:

1. In your PingFederate installation on the source server, open the `pf.jwk` file in the directory:

   `<pf_install>/pingfederate/server/default/data`
2. Copy the key in the file. Ensure you copy the entire key JSON message. For example, you would copy all the bolded text as shown 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 in the directory:

   `<pf_install>/pingfederate/server/default/data`
4. Insert a comma at the end of the last key in the file and append the source key as shown below:

   `{"keys":[{"kty":"oct","kid":"wER9zEpaPe","k":"i0HQr9JmsqjAX4o_BQU1qGJzoLQI-nmwp8u3GyHzTB8"},`**`{"kty":"oct","kid":"j0PUEdAb95","k":"AGi8Lg_ewdl-_30Cx83kDMQE9oNlhgJSa_Pc4I8JTU8"}`**`]}`
5. Save the `pf.jwk` file and start or restart the target server.
6. If applicable, repeat the steps above for each target server running PingFederate.

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 (see *Manage configuration archives* on page 103). 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.

Configuration file copying

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, HyperSQL database when needed.

Certificate management

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 in `<pf_install>/pingfederate/bin/configcopy_templates`.

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 filenames 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 filenames, 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 (see *Authentication* on page 172). 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`, which is 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.

To use configcopy:

1. Ensure access to the Connection Management Web Service is enabled for both the source and target PingFederate servers (see *Authentication* on page 172).
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 preconfigured 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 *Security management* on page 160).

   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 directory:

   `<pf_install>/pingfederate/bin/configcopy_templates`

   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 file.

Outbound provisioning CLI

PingFederate provides as command-line interface (CLI) to help manage automated outbound provisioning at IdP sites (see *Outbound provisioning for IdPs* on page 78). 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 (see *Configure outbound provisioning* on page 295).

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 (see *Configure outbound provisioning settings* on page 140).

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 `bin` directory.

The following tables describes the available global and channel-specific command arguments:

Table 8: Outbound Provisioning CLI Global Options

Command Argument	Description
--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 (see *Review channel settings* on page 304) • User count/dirty-user-record count (e.g.: 5000/12 means 5000 users and 12 dirty records) • Source (as LDAP URL) • Target code
--show-nodes	Shows all the provisioning-server nodes with their status and the last timestamp (applies only to a failover configuration—see the PingFederate *Server clustering guide on page 515*).
--force-node-backup Use with node number: -n <node ID>	Sets the provisioner mode to FAILOVER for the associated PingFederate server node (see the *Server clustering guide on page 515*).

The following table describes the available channel-specific command arguments:

Note: With each command, specify the channel with the argument:

```
-c <channel-id-number>
```
Example:
```
provmgr -c 1 --show-source
```

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

Table 9: Outbound Provisioning CLI Channel-Specific Options

Command Argument	Description
--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.

Command Argument	Description
	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:
-u <provider name>
Or:
-g <LDAP GUID>
	Shows all internal database fields related to the specified user or group, including transitory mapping fields (fields waiting to be pushed to the service provider); for a user, shows all LDAP attributes retrieved from the directory server.
	
	Note: You can obtain user or group names and GUIDs for dirty records, as needed, using any of the `--show-dirty-*` options (described above).
	
	The LDAP GUID, if used and if it is binary, should be entered in hexadecimal format (as shown in log files).
	
	Examples:
	
	`provmgr.sh --show-user -u john@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 is expired (see *Configure outbound provisioning settings* on page 140).

Command Argument	Description
	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 user or to request user input during SSO/SLO processing. These template pages utilize the Velocity template engine, an open-source Apache project, and are located in the `<pf_install>/pingfederate/server/default/conf/template` directory.

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, the PingFederate server supplies values for the Velocity variables used in the template.

Each sample template provided contains specific variables that can be used for rendering the associated Web-browser 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 a second variable available for the page.
`$HttpServletRequest`	A Java object instance of `javax.servlet.http.HttpServletRequest`. Used to add additional knowledge about the request that is otherwise unavailable in the template (for example, the `User-Agent` HTTP header).
`$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 (see *Specify federation information* on page 137).
`$templateMessages`	Used to localize messages in the template (based on user's Locale), an instance of `com.pingidentity.sdk.locale.LanguagePackMessages`. For more information, see the Javadoc for the `LanguagePackMessages` class in the directory `<pf_install>/pingfederate/sdk/doc`. For more information about localization, see *Localization* on page 124.
`$TrackingId`	The user's session tracking ID (see the last *Note* in the section *PingFederate log files* on page 89).

For information about Velocity, please refer to the Velocity project documentation on the Apache web site:

http://velocity.apache.org/engine/releases/velocity-1.4

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 (see tables below). 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 endpoint's `InErrorResource` parameter (see *Application endpoints* on page 442). Administrators can override SSO/SLO success pages by specifying default URLs in the administrative console (for the IdP configuration, see *Configure a default URL and error message* on page 243; for the SP, see *Configure default URLs* on page 315).

The following tables describe each of the templates. To help identify the templates from the end user's point of view, the tables are organized by the titles that appear at the top of the user's browser window.

Note: The titles listed in the tables are the defaults shipped with PingFederate and may be modified. The templates retrieve titles and other text from a language localization file (see *Localization* on page 124).

IdP user-facing pages

Page Title *and template file name*	Purpose	Type	Action
Change Password *html.form.change.password.template.html*	Allows a user to change his or her password via the HTML Form Adapter (see *Configure the HTML Form Adapter* on page 428).	Normal	User input required
Change Password Message *html.form.message.template.html*	Displayed when a user has successfully changed his or her password (see *Configure the HTML Form Adapter* on page 428).	Normal	User input required
Error - Single Logout *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 browser
Error - Single Sign-On *idp.sso.error.page.template.html*	Displayed when IdP-initiated SSO fails on the IdP side and no other SSO error landing page is specified. Displays system errors and information for the user.	Error	Consult log and web developer
IdP Logout *idp.logout.success.page.template.html*	Default page may be displayed when user is logged out of the IdP by the HTTP Basic and HTML Form Adapters (see *Configure the HTTP Basic Adapter* on page 432 and *Configure the HTML Form Adapter* on page 428).	Normal	None
IdP Logout Confirmation *idp.slo.confirm.page.template.html*	Displayed to prompt the user to confirm Single Logout (SLO). For more information, see *Configure a default URL and error message* on page 243).	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 (see *Configure the HTML Form Adapter* on page 428).	Normal	User input required
Please Select Authentication System *sourceid-choose-idp-adapter-form-template.html*	Displayed when the user must choose from several IdP security domains. Based on the user's selection, the server redirects the browser to the appropriate adapter instance for authentication.	Normal	User must make selection
Login Challenge *html.form.login.challenge.*	Displays a configurable challenge form for two-step authentication. This template can be used, for example, to create a RADIUS challenge	Normal	User input required

Page Title *and template file name*	Purpose	Type	Action
template.html	form when using the RADIUS Username/ Password Credential Validator (see *Configure the HTML Form Adapter* on page 428).		
Sign On *html.form.login.template.html*	Displays a configurable user logon form when the HTML Form Adapter is in use for a connection (see *Configure the HTML Form Adapter* on page 428).	Normal	User input required
Signed Out *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 (see *Asynchronous front-channel logout* on page 183).	Normal	None
Signing Out *sourceid-wsfed-idp-signout-cleanup-invisible-template.html*	WS-Federation and OpenID Connect client IdP sign-out processing page. **Note:** No HTML is rendered in the browser.	Normal	None
Success - Single Logout *idp.slo.success.page.template.html*	Displayed when an SLO request succeeds but no other SLO landing page is specified.	Normal	None
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
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:** Normally not displayed if Javascript executes properly.	Normal	None

SP user-facing pages

Page Title *and template file name*	Purpose	Type	Action
Account Link Removed *TerminateAccountLinks.page.template.html*	Communicates a user's successful "defederation" operation.	Normal	None
Account Linking *LocalIdPasswordLookup.form.template.html*	Used to authenticate a user at the SP when an account link needs to be established.	Normal	None
Authentication Failed *sourceid-wsfed-idp-exception-template.html*	Displayed when an authentication challenge fails during WS-Federation processing.	Error	Consult log and web developer

Page Title *and template file name*	Purpose	Type	Action
Error - Single Logout *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
Error - Single Sign-On *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
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
Signed Out - Service Provider *sourceid-wsfed-sp-signout-cleanup-template.html*	Displays the user's sign-out status.	Normal	None
Success - Single Logout *sp.slo.success.page.template.html*	Displayed when an SLO request succeeds and no other SLO success landing page is specified.	Normal	None
Single Sign-On Target Unspecified *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 (see *Configure default URLs* on page 315).	Error	Consult web developer, or specify default URL

Either IdP or SP user-facing pages

Page Title *and template file name*	Purpose	Type	Action
Error - Single Sign-On *generic.error.msg.page.template.html*	For an Auto-Connect SSO transaction, indicates a range of possible error conditions (see *Using Auto-Connect* on page 75): • The requesting Auto-Connect partner is not found in the PingFederate server's list of allowed domains. • The partner's metadata is not accessible. • The server is not configured for Auto-Connect. • General error, with error code.	Error	Consult log, check configuration, or contact partner. If unresolved, contact Ping Identity *support*.
HTTP Error *http.error.page.template.html*	Indicates that an HTTP error has occurred and provides the HTTP status code.	Error	Consult log, contact PingIdentity *support*
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*

Page Title *and template file name*	Purpose	Type	Action
Multiple SSO in Progress *speed.bump.template.html*	Displayed to a user when response is slow due to simultaneous SSO requests coming from multiple browser tabs.	Normal	None
Page Expired *state.not.found.error.page.template.html*	Displayed when simultaneous SSO requests from multiple tabs using the same key value cause a user session to be overwritten or deleted and remaining requests attempt to retrieve the state fail.	Error	None
Sign On *AbstractPasswordIdpAuthnAdapter.form.template.html*	Challenges user for credentials when authentication can take place via HTTP Basic Authentication or an HTML form, depending on the operational mode.	Normal	User must sign on
Submit Form *form.autopost.template.html*	Whenever the server posts a form, this template is used to auto-submit the form. If Javascript is disabled, the user is prompted to click a button to post the form manually. **Note:** Normally not displayed if Javascript executes properly.	Normal	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
Access Revocation `oauth.access.grants.page.template.html`	Provides a means for the end users (resource owners) to revoke persistent access grants.	Normal	User input required
Information Access 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 Settings** > **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

Localization

The English display text in each of the user-facing templates is retrieved from a default localization file, `pingfederate-messages.properties`, in the directory:

`<pf_install>/pingfederate/server/default/conf/language-packs`

An administrator can localize the text by copying the default file, providing translated text in place of English, and then adding the standard language tag to the base filename (as indicated in browser settings).

For example, for text in French rename the translated copy of the localization file in the language-packs directory to:

`pingfederate-messages_fr.properties`

To include a region, append the capitalized abbreviation using an *underscore*.

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 (see *Retrieving localized messages* on page 125).

For example, for Canadian French:

`pingfederate-messages_fr_CA.properties`

Note: If the system language of the PingFederate server is not English, copy pingfederate-messages.properties as `pingfederate-messages_en.properties` for the English users (if any), translate `pingfederate-messages.properties` for the local users, and provide additional translations as needed.

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 templates 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 `cache-language-pack-messages` to false in:

`.../pingfederate/server/default/data/config-store/locale-options.xml`

Be sure to return the value to `true` when testing is complete. Note that restarting the server is required for changes to configuration files.

Overriding locales with 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, PingFederate supports overriding the locale via a cookie named `pf-accept-language`. The cookie value must conform to guidelines defined under *IETF BCP 47*. For more information, see documentation for the Java core method PingFederate uses to parse the cookie: *`Locale.forLanguageTag(String languageTag)`* (`docs.oracle.com/javase/7/docs/api/java/util/Locale.html#forLanguageTag(java.lang.String)`). (This locale-override behavior is the default implementation of the Java interface `LocaleOverrideService` defined in the PingFederate SDK. For more information, see the Javadoc for that interface in the directory `<pf_install>/pingfederate/sdk/doc`.)

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.

Retrieving 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 directory `<pf_install>/pingfederate/sdk/doc`.

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 `<pf_install>/pingfederate/server/default/data/config-store/password-rules.xml` file.
2. Save the changes.
3. Restart PingFederate.

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

Manage expired persistent grants

PingFederate® automatically removes expired persistent grants once a day in batches of 500. If expired grants are growing rapidly, you may increase the frequency of the cleanup process, the number of expired persistent grants to be removed per batch, or both.

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 database server. We recommend making changes gradually to observe the impact, if any.

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/timer-intervals.xml` file.
2. Optional: Update the AccessGrantCleanerInterval value to adjust the frequency of the cleanup process.
3. Optional: Update the ExpiredGrantBatchSize value to adjust the number of expired grants to be removed per batch.
4. Save your change.
5. 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.

   The cleanup process only runs on the console node.

Extend the lifetime of the PF cookie

PingFederate identifies sessions by their respective PF cookie. By default, the PF cookie is a session cookie. You can extend the lifetime of the PF cookie by making it a persistent cookie. Unlike session cookies, persistent cookies are saved to disk, enabling your web browser to reuse them when restarted.

1. Edit the `<pf_install>/pingfederate/server/default/data/config-store/session-cookie-config.xml` file.

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.

   Note: It is not necessary to restart PingFederate on any running engine node.

The HTML Form Adapter utilizes the PF cookie to manage its sessions. For more information, see *Configure the HTML Form Adapter* on page 428.

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 Strict-Transport-Security 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) 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, and then click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen.

   Note: For each engine node, restart PingFederate to reload the `response-header-admin-config.xml` or `response-header-runtime-config.xml` file (or both) after the configuration is replicated.

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 (see *System-services endpoints* on page 452). 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 `<pf_install>/pingfederate/server/default/conf/template/heartbeat.page.template` Velocity template file to specify the desired information to be returned by the heartbeat endpoint. (An inline sample is provided. Template customization does not require a restart of PingFederate.)

   For a clustered PingFederate environment, repeat these steps on each engine node.

Customize the favicon for application and protocol endpoints

PingFederate provides a favorite icon (favicon) for its Application and Protocol Endpoints (see *Application endpoints* on page 442, *View IdP protocol endpoints* on page 244 for IdP and *View SP protocol endpoints* on page 317).

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 engine 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 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.

Note: It is not necessary to restart PingFederate on any running engine node.

For more information, see *Attribute mapping with multiple data sources* on page 494.

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 129
- *Manage data stores* on page 143
- *IdP discovery* on page 155
- *Configure redirect validation* on page 158

Note: The information in this chapter is presented from the viewpoint of an administrative user with "Admin" permissions (see *Account management* on page 105).

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* chapter of the *Get started with PingFederate* guide).

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 (see *Start PingFederate for the first time* in the "Installation" chapter of the *Get started with PingFederate* guide), but you can change or add to them as needed from the Main Menu.

Note: For information about WS-Trust STS Settings, see *Configure STS authentication* on page 379.

Information in this section covers:

- *Set administration options* on page 130
- *Enter system information* on page 130
- *Configure runtime notifications* on page 130
- *Configure runtime reporting* on page 131
- *Manage PingOne settings* on page 134
- *Managing accounts* on page 136
- *Choose roles and protocols* on page 136
- *Specify federation information* on page 137
- *Configure system options* on page 138
- *Configure outbound provisioning settings* on page 140
- *Configure metadata signing* on page 141
- *Configure metadata lifetime* on page 141

- *Edit and save server settings* on page 142

Set administration options

On the System Administration screen, PingFederate provides a choice of single- or multi-user access to the administrative console.

Note: If you are using your network's LDAP user-data store or client certificates for administrative-console authentication, this screen is not presented (see *Alternative console authentication* on page 108).

If you choose Single-user Administration, the console is accessible only by using the default `Administrator` ID, for which full privileges are provided. Multi-user Administration (the default) provides role-based access control (see *Account management* on page 105).

Tip: To return to single-user administration after having previously enabled multi-user, only one user can be marked as active under Account Management.

Enter system information

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

To reach this screen

1. Click **Server Configuration** on the Main Menu.
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. In 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 89).

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 110).

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) (see *Runtime monitoring using JMX* on page 131).

Using SNMP monitoring

The SNMP Management Information Base (MIB) defines network data available for SNMP monitoring. The MIB file is located in: `<pf_install>/pingfederate/SNMP`

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).

Configure access to SNMP monitoring on the Runtime Reporting screen.

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 (for each engine separately when PingFederate is deployed in a cluster—see the PingFederate *Server clustering guide on page 515*).

Gets

PingFederate responds to two SSO/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 `pingfederate/SNMP` directory. (For more information about MBeans, see the next section, *Runtime monitoring using JMX* on page 131).

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.

- If you configure Traps, change settings as needed and then click **Test SNMP Configuration** to send a single Trap to your network-management console.

  You can also use an HTTP call at any time to verify that the PingFederate server is running (see *System-services endpoints* on page 452).

To reach this screen:

1. Click **Server Configuration** on the Main Menu.
2. Click **Server Settings** under System Settings.
3. Click **Runtime Reporting** on the Summary screen.

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.

PingFederate's 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 (see *Example Jetty metrics* on page 133).

Important: Authentication is required for JMX-client access to PingFederate runtime data (see *Authentication* on page 172).

Tip: Administrators can supplement JMX monitoring information by applying third-party analysis and reporting tools to the security audit log (see *Security audit logging* on page 93). That log records fine-grain details (including, for example, response times and event types) for all server transactions.

SSO-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` (see the previous section, *Configure runtime reporting* on page 131).

Provisioning monitoring

Important: Monitoring Outbound Provisioning transactions using JMX has been deprecated. By default, PingFederate logs outbound provisioning events to `provisioner-audit.log` (see *Outbound provisioning audit logging* on page 95).

To re-enable JMX monitoring:

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

Note: Repeat these steps on each PingFederate server configured to process Outbound Provisioning (see *Outbound provisioning for IdPs* on page 78).

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.

Table 10: Outbound Provisioning JMX Monitoring Options

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.

Method	Description	Parameter
		`fromDate` – See Note below. `toDate` – See Note below.
`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 Note below.
`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 Note below.
`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 empty string for a date parameter indicates no constraint for that end of the range.

Example 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 described in previous sections.

Table 11: Selected Jetty Metrics

MBean	Attributes

`org.eclipse.jetty.server:context=0,type=serverconnector,id=0`
 	
`connectorstatistics`
 	
(For Jetty connectors including the primary and
	`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.

MBean	Attributes
secondary PingFederate runtime server ports)	`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`
`java.lang: MemoryPool`
`java.lang: GarbageCollection`
`java.lang: OperatingSystem`
	Various attributes measuring CPU usage and memory.

Advanced JMX configuration

By default, PingFederate uses port 1099 for its JMX server. To change the port or other JMS configuration items, if needed, modify the configuration file `jmx-remote-config.xml` in the directory `<pf_install>/server/default/conf`.

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 161). (This should be a consideration only during testing, when using the certificate installed with PingFederate or another self-signed certificate.)

Manage PingOne settings

After connecting PingFederate to PingOne through a managed SP connection, use the **Server Configuration** > **Server Settings** > **PingOne Settings** screen to manage the following options:

- *SSO from PingOne to the PingFederate administrative console*
- *Monitoring of PingFederate from PingOne*

PingFederate also automatically rotates the signing certificate used by the managed SP connection, see *Managed SP connection to PingOne and signing certificate* on page 168.

SSO from PingOne to the PingFederate administrative console

In PingFederate 8.0 (or higher), 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 subsequently decide to disable the SSO from PingOne option, 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 setup PingFederate without PingOne at the beginning but have subsequently created a managed SP connection to PingOne using the Connect to PingOne wizard, use the 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. (For information about native authentication and alternative console authentication, see *Account management* on page 105 and *Alternative console authentication* on page 108, respectively.) 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:

1. Verify the feature is enabled in the PingOne Settings screen.
2. Launch your browser.
3. Go to:

   `https://<DNS_NAME>:9999/pingfederate/app`

   where `<DNS_NAME>` is the fully qualified name of the machine running the PingFederate server.

   Note: The port number 9999 is set by default. For information on changing this setting, see *PingFederate properties* on page 112 in the "System Administration" chapter of the PingFederate *Administrator's manual*.

If you have already logged on to the PingOne admin portal, the Main Menu opens. 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 Main Menu opens.

To sign on via native or alternative console authentication:

When the SSO from PingOne feature is enabled, you can use the direct login page to sign on via native authentication or alternative console authentication.

1. Launch your browser.
2. Go to:

   `https://<DNS_NAME>:9999/pingfederate/app?service=page/directLogin`

   where `<DNS_NAME>` is the fully qualified name of the machine running the PingFederate server.

   Note: The port number 9999 is set by default. For information on changing this setting, see *PingFederate properties* on page 112 in the "System Administration" chapter of the PingFederate *Administrator's manual*.

To disable native and alternative console authentication:

1. Edit `<pf_install>/pingfederate/bin/run.properties`.
2. Change the `pf.console.authentication` value to `none`.
3. Save the change.
4. Restart PingFederate.

   Note: In a clustered PingFederate environment, you only need to modify `run.properties` 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://<DNS_NAME>:9999/pingfederate/app`.

If you wish to re-enable native or alternative console authentication, update the `pf.console.authentication` setting 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 wish to monitor PingFederate from the PingOne admin portal, clear the option check box and click **Save** in the PingOne Settings screen.

Managing accounts

When you choose multi-user system administration, you can create users during installation or while configuring Server Settings (see *Set administration options* on page 130).

Note: If you are using your network's LDAP user-data store or client certificates for PingFederate authentication, the Account Management screen is not presented (see *Alternative console authentication* on page 108).

Alternatively, you can set up and maintain user accounts later as a separate task (assuming you have user administration permissions—see *Account management* on page 105). By default for installation, the user "Administrator" has full system permissions.

- To continue, click **Next** or **Save**.
- For information about adding or managing users, see *Account management* on page 105.

Choose roles and protocols

On the Roles and Protocols screen, select which role(s) your organization plays and which sets of standards you will use with your PingFederate server (see *Supported Standards* chapter of the *Get started with PingFederate* guide).

Note: If you are using the PingFederate WS-Trust STS for either an IdP, an SP, or both, notice that a new configuration step, WS-Trust STS Settings, appears under the Server Settings tab. For information about this configuration, see *WS-Trust STS configuration* on page 378.

Also on this screen, you can choose any of several options:

- Enable the PingFederate OAuth AS (see *PingFederate OAuth AS* on page 60).

  You can also extend OAuth processing to OpenID Connect policy configurations (see *OpenID Connect* on page 64).
- As an SP, if you are using SAML 2.0 XASP for multiple IdP connections, you may choose to have PingFederate determine dynamically which connection to use (see *Attribute Query and XASP* in the "Supported Standards" chapter of the *Get started with PingFederate* guide)
- For either role you can enable Auto-Connect for SAML 2.0 connections (see *Using Auto-Connect* on page 75).
- Also for either role, you can enable the Outbound/Inbound Provisioning option (see *User provisioning* on page 78).
- If you are installing PingFederate and are not sure of your selections, just click **Next**.

  Note: If you do not choose a role during installation, you must return to this screen to do so before you can configure connections to partners.

To reach this screen for editing:

1. Click **Server Configuration** on the Main Menu.
2. Click **Server Settings** under System Settings.
3. Click **Roles and Protocols** on the Summary screen.

To choose roles and protocols:

1. Select your federation role(s) and then select at least one protocol.

   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.)

2. Optional: If you are using SAML 2.0 and want to configure Auto-Connect, select that feature for your role(s) (see *Using Auto-Connect* on page 75).

   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 (accessible from the Main Menu) for your respective role.

   When you make this selection, two additional steps are added to the System Settings task:

   - Metadata Signing (see *Configure metadata signing* on page 141)
   - Metadata Lifetime (see *Configure metadata lifetime* on page 141)

3. Optional: If you are using PingFederate as an IdP for provisioning or have installed a SaaS Connector package, select the Outbound Provisioning check box.

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

   (For conceptual information about outbound provisioning, see *Outbound provisioning for IdPs* on page 78.)

   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 (see *Review channel settings* on page 304). Alternatively, you can deactivate the associated SP connection; note, however, that this will also disable SSO/SLO transactions (see *Review an SP connection settings* on page 304).

4. Optional: If you are using PingFederate as an SP for provisioning, select Inbound Provisioning.

   (For conceptual information about inbound provisioning, see *Provisioning for SPs* on page 79.)

5. Optional: If you are using SAML 2.0 XASP as an SP for multiple IdP connections, you may select the option to determine dynamically which connection to use, based on the X.509 certificate presented (see *Manage attribute requester mappings* on page 316).

   Tip: After you make this selection and create XASP IdP connections (see *Configure the Attribute Query profile* on page 285), configure dynamic IdP discovery via the **Attribute Requester Mapping** link on the SP Configuration 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* in the “Supported Standards” chapter of the *Get started with PingFederate* guide.

6. Click **Next** (or **Save**, if you are modifying existing selections).

For information about configuring settings associated with your selections, see these relevant portions of this manual:

- *OAuth configuration* on page 181
- *Identity provider SSO configuration* on page 238
- *Service provider SSO configuration* on page 308
- *WS-Trust STS configuration* on page 378
- *IdP discovery* on page 155

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 *Using Auto-Connect* on page 75).

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 101).
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.
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 77). 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 *Using Auto-Connect* on page 75).
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 136).

To reach this screen:

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

Configure system options

The 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

To reach this screen:

1. Click **Server Configuration** on the Main Menu.

2. Click **Server Settings** under System Settings.
3. Click **System Options** on the Summary screen.

Disable automatic connection validation

Automatic multi-connection error checking occurs by default for all configured connections whenever you access connection lists (via the Manage Connections screen for SP or IdP connections). The same multi-connection checking also occurs when you access the Manage IdP/SP Adapter Instances screens.

Because validation time increases with the number of connections and adapter instances, this option is provided in case you experience any noticeable delays in loading either of the Manage Connection screens or the adapter management screens. Disabling the feature results in immediate display of the screens, deferring error checking to manual controls on Manage Connection screens.

Note: This option does not affect the validation of connections as they are being configured or modified. Also, individual connections are always validated automatically when accessed for editing, regardless of the setting on this screen.

The multi-connection error checking is intended to verify that completed connections have not been affected by any subsequent changes in adapter configurations or other dependencies such as data-store access (see *Data stores* on page 70).

For more information about connection validation, see:

- *Manage SP connection validation* on page 249(for SP connections)
- *Manage IdP connection validation* on page 321 (for IdP connections)

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. 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 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 HTTP Header for Client IP Addresses, 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 HTTP Header for Hostname, 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 for either WS-Trust STS authentication or SAML back-channel 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 field 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, you can select the database that PingFederate uses 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 HyperSQL 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 HyperSQL, Oracle database, Oracle MySQL, and Microsoft SQL Server as internal provisioning data stores. However, any relational database should work as well; adaptable

setup scripts used for HyperSQL, Oracle database, Oracle MySQL, and Microsoft SQL Server 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** role is enabled on the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

1. Select a data store from the drop-down 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 (see *Manage data stores*).
2. Optional: Change the **Synchronization Frequency** value.

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** workflow 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 drop-down list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing and decryption keys and certificates* on page 164).
2. Optional: Select the Signing Algorithm from the drop-down 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 77).

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 *Connections Management for IdP* and *Import SP metadata* on page 251, or *Connections Management for SP* and *Import IdP metadata* on page 323.

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 *Ping Identity's web site*.

Note: A managed SP connection to PingOne is a connection created by the initial setup wizard or the Connect to PingOne wizard in PingFederate 8.0 (or higher).

If an SP connection to PingOne was created in PingFederate 7.3 (or older), you can delete that connection and let the Connect to PingOne wizard create a new managed SP connection to PingOne for you. The benefits of a managed connection include the capability to SSO from PingOne to the PingFederate administrative console, monitoring of PingFederate from PingOne, automatic metadata exchange between PingFederate and PingOne, and automatic certificate rotations (see *Manage PingOne settings* on page 134 and *Managed SP connection to PingOne and signing certificate* on page 168).

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.

Connect with a new AD configuration

When selected, the Connect to PingOne 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
- Enabling users and group provisioning to PingOne
- Enabling the capability to SSO from PingOne to the PingFederate administrative console

Connect to an existing AD or other user store

When selected, the SP connection wizard initiates a new managed SP connection to PingOne for you. The Attribute Contract is 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.

   Tip: If you use the HTML Form Adapter created by the initial setup wizard (in PingFederate 8.0 or newer), you can map the six attributes from the adapter contract. No data store lookup is required.

   Note: If you need to retrieve objectGUID from an LDAP data store, configure objectGUID to be handled as binary data in the LDAP data store and set **Attribute Encoding Type** to `Hex` in the **LDAP Binary Attribute Encoding Types** screen (see *Specify LDAP binary attributes* on page 148 and *Specify encoding for LDAP binary attributes* on page 271 in the PingFederate *Administrator's manual*).
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 outbound SaaS or SCIM provisioning, Browser SSO, WS-Trust STS, and OAuth transactions. It can also read and write to data stores for inbound just-in-time or SCIM provisioning.

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 Connector, 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.

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. Standard data stores include JDBC-enabled databases and LDAP v3-compliant directories. Alternatively, you can develop a driver using the PingFederate Custom Source SDK to connect to non-JDBC databases.

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

2. In 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 wizard to configure 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: Ensure that your JDBC 4.1 (or higher) database-driver JAR file is installed in the `<pf_install>/pingfederate/server/default/lib` directory and that the driver is compatible with the Server JRE version in use and with the target database. 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, in the format:
	`jdbc:mysql://databaseservername/databasename`
	`jdbc:sqlserver://databaseservername;databaseName=databasename`
	`jdbc:oracle:thin:@databaseservername:databasename`
	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 if 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 examples: • `com.microsoft.sqlserver.jdbc.SQLServerDriver` • `com.mysql.jdbc.Driver` • `oracle.jdbc.OracleDriver` The drive class name should be supplied by the database software vendor in a JAR file.
Username	The name that identifies the user when connecting to the database.

Field	Description
(Required)	
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 examples: • `SELECT 1 from dual` (for Oracle Database or Oracle MySQL) • `SELECT getdate()` (for Microsoft SQL Server) **Tip:** To use this feature for Oracle MySQL, you must also add a 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 record from a column and return the results as a multi-value 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. 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

You configure access to user attributes on your LDAP server by providing basic LDAP information.

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**.

Task	Steps
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 your JDBC connection, as described in the following table.

Field	Description
Hostname(s) (Required)	The DNS name or IP address of the data store, which may include a port number; example: `181.20.42.130:389`. For failover, you can enter multiple LDAP servers, each separated by a space. **Note:** If multiple LDAP servers are specified, each server must be accessible using the same user DN and password (unless the **Bind Anonymously** check box is selected).
LDAP Type (Required)	If you are using this data store for outbound provisioning and your LDAP store is Microsoft Active Directory (AD), Oracle Directory Server, or UnboundID Data Store, select the applicable type from the list , such that PingFederate® can pre-configure many provisioning settings automatically. **Tip:** If you are using outbound provisioning and your LDAP server is *not* AD or Oracle, you may wish to define a custom LDAP Type to streamline the provisioning configuration (see *Modify source settings*). 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 LDAP directory server supports anonymous binding and if no credentials are needed to access the data store. **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 LDAP directory server likely requires a specific service account to handle the communication between PingFederate and the target data store. When selected, user DN and password are not required.
User DN	The username credential required to access the data store. **Important:** The service account must have permission to search the directory for user-account information. If your use cases involve reading from the LDAP 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 LDAP server running on Microsoft Windows Server 2008 R2 (or newer), enter an AD User account; do not use a Computer account.
Password	The password credential required to access the data store.
Use LDAPS	Connects to the LDAP data store using LDAPS. This selection applies equally to any secondary servers specified in the **Hostname(s)** field.

Field	Description
	Important: We recommend that all LDAP connections be secured by using LDAPS.
Mask Values in Log (Checkbox)	Determines whether all attribute values returned from this data store should be masked in PingFederate log files.

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

Set advanced LDAP options

PingFederate® maintains a search pool and a bind pool for each LDAP data store instance 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 for the related data store. These settings are applicable to both the search pool and the bind pool.

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

   Tip: The default values are conservative based on the default thread-pooling limits allowed by the installed PingFederate web container. (These limits are 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.
Test Connection on Return	Indicates whether objects are validated before being returned to the pool.
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.
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.
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	The largest number of active connections that can remain in each pool (not including the temporary connections that are managed automatically when the

Field	Description
(Required)	**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.

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 that 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 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 user-management LDAP server is not Microsoft Active Directory or Oracle Directory Server, you can define a custom LDAP type for PingFederate® to use to streamline the provisioning configuration.

When the LDAP server is defined, its type appears in the **LDAP Type** list on the **LDAP Configuration** screen.

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.

   The template.name 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 LDAP server.

   These are properties from the outbound provisioning setup (see *Modify source settings*).
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.

5. Go to the **Server Configuration** > **Data Store** screen.

   You may create a new LDAP connection to your LDAP directory server by using the newly defined custom LDAP type. (Select the custom LDAP type from the list in the **Data Store Type** 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. (For more information, see the *SDK developer's guide*).

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.

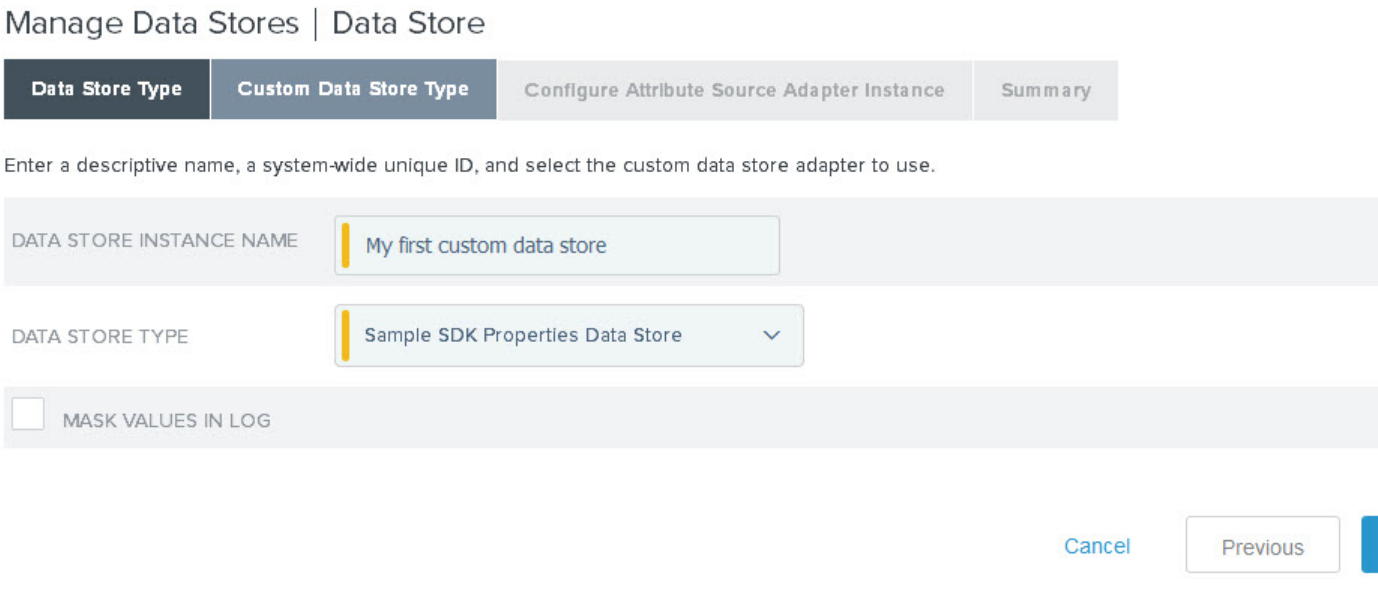
Task	Steps
Configure a new data store	1. On the **Manage Data Stores** screen, click **Add New Data Store**. 2. On the **Data Store Type** screen, select **Custom**.
Modify an existing data store	1. On the **Manage Data Stores** screen, click the applicable data store by its name. 2. On the **Summary**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

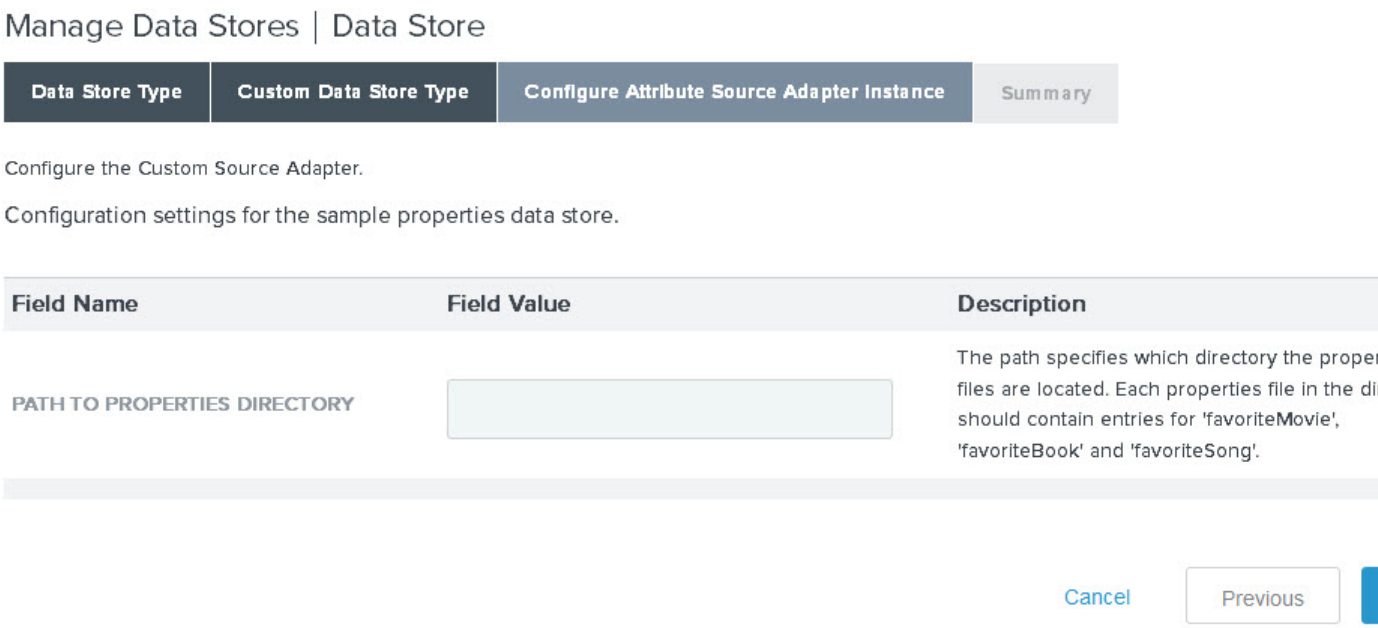
      If the list is empty, verify that the data-store driver is implemented and installed successfully.
   c) Select the **Mask Values in Log** check box if all attribute values returned from this data store should be masked in PingFederate log files.

   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**:



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.



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 HyperSQL database as the account-link repository (see *Account linking* on page 67). 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 515*). 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 HyperSQL 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 144).

Be sure to save the configuration on the Manage Data Stores screen.

2. Any time after saving the database connection, return to the Manage Data Stores screen from the Main Menu.

   (To reach the screen, click **Data Stores** under System Settings.)

3. Copy the System ID for the database you want.
4. In the directory `<pf_install>/pingfederate/server/default/data/config-store`, open the file:

   `org.sourceid.saml20.service.impl.AccountLinkingServiceDBImpl.xml`

   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 PingFederate *Server clustering guide on page 515*.

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 145).

   Be sure to save the configuration on the Manage Data Stores screen.

2. Any time after saving the LDAP data store connection, return to the Manage Data Stores screen from the Main Menu.

   (To reach the screen, click **Data Stores** under System Settings.)

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 multi-valued string attribute on a user object class. We recommend that you extend the LDAP schema with a custom attribute for use here. See this *MSDN article* 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 PingFederate *Server clustering guide on page 515*)

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 HyperSQL database by default to maintain persistent grants (if any) for the OAuth AS. (Persistent grants can result from some OAuth use cases, see *Persistent vs. transient grants* on page 62 for more information).

Important: When persistent grants are expected, administrators should consider changing this configuration to use an external secured database or an LDAP directory server for production standalone deployments. For server clustering, an external database or an LDAP directory server is required, because the HyperSQL 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 database server or LDAP directory server and modifying a couple PingFederate configuration XML files.

PingFederate also supports other storage solutions through the PingFederate SDK. The process involves implementing the `AccessGrantManager` interface and modifying a PingFederate configuration XML file to point to the custom class.

To create the database tables for grant storage:

Run the table-setup scripts for your database server provided in the `<pf_install>/pingfederate/server/default/conf/access-grant/sql-scripts` directory.

Note: If scripts are not provided for your database server, you can derive the setup from information available in any of the other scripts.

To change the 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 144).
2. Go to the **Server Configuration** > **Data Stores** screen.
3. Copy the **System ID** value for the database you want.
4. Edit the `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

   Change the value of the item named `PingFederateDSJNDIName` to the **System ID** value you copied, and then save the file.

   Note: For a clustered PingFederate environment, edit the `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file on the console node.
5. Edit the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.
   a) Locate the `AccessGrantManager` block:

```
<!-- 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.

Important: For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

6. Start or restart PingFederate.

   For a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen to push the modified `org.sourceid.oauth20.token.AccessGrantManagerJdbcImpl.xml` file to all engine nodes, and then restart 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 persistent grants* on page 126.

To prepare your LDAP directory server for grant storage:

Specific schema objects are required in order for PingFederate to store grants on your LDAP directory server. LDIF scripts are provided for supported LDAP directory servers (see *System requirements* on page 13).

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.

To store grants in an LDAP directory store:

1. If you have not already done so, create a connection to your LDAP directory server (see *Configure an LDAP connection* on page 145).
2. Go to the **Server Configuration** > **Data Stores** screen.
3. Copy the **System ID** value for the LDAP data store connecting to your LDAP directory server.
4. Edit a configuration file in the `<pf_install>/pingfederate/server/default/data/config-store` directory.

LDAP directory server	Configuration file
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl.xml`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl.`
UnboundID Data Store	`org.sourceid.oauth20.token.AccessGrantManagerLDAPUnboundIDIm`

   a) Use the **System ID** value you copied as the value of the item named `PingFederateDSJNDIName`.
   b) Enter a value for the item named `SearchBase`.

   Tip: This is the LDAP search base that points to the `accessGrants` location. For more information, see 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.

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

5. Edit the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.
   a) Locate the `AccessGrantManager` block:

```
<!-- 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 one of the following values:

LDAP directory server	Class value
Microsoft Active Directory	`org.sourceid.oauth20.token.AccessGrantManagerLDAPADImpl`
Oracle Directory Server Enterprise Edition	`org.sourceid.oauth20.token.AccessGrantManagerLDAPOracleImpl`
UnboundID Data Store	`org.sourceid.oauth20.token.AccessGrantManagerLDAPUnboundID`

c) Save the file.

Important: For a clustered PingFederate environment, edit the `hivemodule.xml` file on each node.

6. Start or restart PingFederate.

   For a clustered PingFederate environment, click **Replicate Configuration** on the **Server Configuration** > **Cluster Management** screen to push the modified configuration file to all engine nodes, and then restart each engine node to activate the changes.

To support a custom storage solution for grant storage:

1. Implement the `AccessGrantManager` interface.

   Note: For more information, refer to the Javadoc for PingFederate located in the `<pf_install>/pingfederate/sdk/doc` directory and the `SampleAccessGrant.java` sample implementation in the `<pf_install>/pingfederate/sdk/plugin-src/access-grant-example/java/com/pingidentity/accessgrant/` directory.

2. Edit the `hivemodule.xml` file in the `<pf_install>/pingfederate/server/default/conf/META-INF` directory.

   a) Locate the `AccessGrantManager` block:

```
<!-- 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 the name of your class.

   c) Save the file.

   Important: 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 PingFederate.

Define an OAuth client data store

By default, PingFederate stores OAuth clients entered into the system through the administrative console (see *Manage OAuth clients* on page 194) or the REST-based application programming interface (see *PingFederate administrative API* on page 480) in XML files.

Alternatively, it can be configured to use a database for OAuth client records, allowing the records to be managed via the administrative console, the REST-based application programming interface, or an OAuth Client Web Service (see *OAuth Client Management Service* on page 471). When database usage is enabled, PingFederate uses its internal HyperSQL database by default. Administrators should consider changing this configuration to use an external secured database for production standalone deployments.

Important: For server clustering, an external database *is* required, because the HyperSQL database cannot be shared across other PingFederate engine nodes.

Enabling database usage and changing the default database involve modifying two PingFederate configuration files and restarting the server. For production databases, an administrator must also create a database table for OAuth clients using an SQL script.

To enable usage of an OAuth-client database:

1. In the directory `<pf_install>/pingfederate/server/default/conf/META-INF`, open the file:
   `hivemodule.xml`
2. Search the file for:
   `ClientManagerXmlFileImpl`
   Replace with:
   `ClientManagerJdbcImpl`
   You can reverse the setting if you ever want to revert to the default XML file storage implementation.

   Caution: If clients are already defined using the default XML implementation, the records are not moved to a database: they must be recreated. Likewise, if the configuration is changed back to the default, database records are not ported to XML files.
3. Restart PingFederate.
4. If you are running PingFederate in a server cluster, repeat the previous steps for each server node.
5. If you are ready to define an external enterprise database, follow the steps below.

To change the database:

1. For the database you want to use, run one of the table-setup scripts provided in the directory: `<pf_install>/pingfederate/server/default/conf/oauth-client-management/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.
2. If you have not already done so, create a connection to the database (see *Configure a JDBC database connection* on page 144).
   Be sure to save the configuration on the Manage Data Stores screen.
3. Any time after saving the database connection, return to the Manage Data Stores screen from the Main Menu.
   (To reach the screen, click **Data Stores** under System Settings.)
4. Copy the System ID for the database you want.
5. In the directory `<pf_install>/pingfederate/server/default/data/config-store`, open the file: `org.sourceid.oauth20.domain.ClientManagerJdbcImpl.xml`
   Change the value of the `item` named `PingFederateDSJNDIName` to the System ID you copied in the previous step and save the file.

   Note: If you are running PingFederate in a cluster, be sure to make this modification on the administrative server.
6. Start or restart PingFederate.

   Note: If you are running PingFederate in a cluster, push the new configuration to other server nodes (see the PingFederate *Server clustering guide on page 515*).

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: `24,855` 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 PingFederate endpoint intended for that purpose (see `/idp/writecdc.ping` in *IdP endpoints*).
   - 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.

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 well-formed SSO or SLO links 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 redirection to a malicious web site.

To prevent such an attack, PingFederate provides a means of validating SSO and SLO transactions to ensure that the designated target resource exists in a domain controlled by the SP through a list of configurable URLs.

Note: Validating target resource for SSO is applicable for IdP connections, adapter-to-adapter mappings and SAML 2.0 IdP Discovery.

The following default target URLs are always allowed:

- The default target URL for any IdP connections (see *Configure default target URLs (optional)* on page 351)
- The default target URL for any adapter-to-adapter mappings (see *Configure a default target URL (optional)* on page 419)
- The SP default URL for successful SSO (see *Configure default URLs* on page 315)
- The IdP default URL for successful SLO (see *Configure a default URL and error message* on page 243)

They do not need to be entered into the list manually.

Tip: PingFederate is also capable of validating the error resource (`InErrorResource`) parameter for the same reason. For more information, see *IdP endpoints* on page 442, *SP endpoints* on page 444 and *System-services endpoints* on page 452.

Important: PingFederate enables target resource validation for SSO and SLO, as well as error resource URLs, and requires HTTPS in new installations by default.

For backward compatibility, PingFederate Upgrade Utility does not enable these options if they were not selected in the previous PingFederate installation. Although optional, it is strongly recommended to enable target and error resource validation (including the HTTPS option) and to enter all expected resources to prevent unauthorized access.

To reach this screen:

1. Click **Server Configuration** on the Main Menu.
2. Click **Redirect Validation** under System Settings.

To enable target resource validation for SSO and/or SLO:

- Select the SSO and/or SLO check box(es).

To enable error resource validation:

- Select the Enable InErrorResource validation check box.

To enter expected resources:

1. Indicate whether to require HTTPS.

   Important: This selection is recommended to ensure that target validation will always prevent message interception for this type of potential attack, under all conceivable permutations.

2. Enter the domain or IP address containing the expected target resource under Valid Domain Name.

   Use the domain only, without qualifiers. For example:

   ```
   mycompany.com
   ```

Using an initial wildcard and period for a domain name will cover multiple subdomains. For example:

`*.mycompany.com`

covers `hr.mycompany.com` or `email.mycompany.com`. Note that `*.mycompany.com` does not cover `mycompany.com` itself.

(Optional) Enter the exact path of the resource (case-sensitive) 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. For example:

`/inbound/Consumer.jsp`

allows `/inbound/Consumer.jsp` but rejects `/inbound/consumer.jsp`

Tip: You can also enter multiple query parameters with or without a fragment.

For example: `/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` (Optional) Select the Allow Any Query / Fragment check box if you want to allow any query parameters or fragment in the resource.

Note: When selected, no query parameter and/or fragment is allowed in the path.

Select the TargetResource for SSO, TargetResource for SLO and/or InError Resource check box(es) to indicate the resource type(s).

Click **Add**.

3. Repeat the previous step to add additional entries as needed.
4. 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 whitelist for the parameter is managed within the connection settings 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 in new installations by default.

For backward compatibility, PingFederate Upgrade Utility does 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 IdP connection to prevent unauthorized access (see *Specify a service URL (WS-Federation)* on page 347).

To reach this screen:

1. Click **Server Configuration** on the Main Menu.
2. Click **Redirect Validation** under System Settings.
3. Click **Partner Redirect Validation**.

To enable wreply validation for SLO:

1. Select the Enable wreply validation for SLO check box.
2. Click **Save**.

To disable wreply validation for SLO:

1. Clear the Enable wreply validation for SLO check box.
2. Click **Save**.

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 160
- *Authentication* on page 172

Note: This information is presented from the viewpoint of an administrative user with "Crypto Admin" permissions (see *Account management* on page 105).

Certificate management

PingFederate administrators manage certificates via the Certificate Management section on the Server Configuration menu.

Manage trusted certificate authorities

You can import your federation partner's CA certificate or self-signed certificate(s) into PingFederate's global trust list. 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. Click **Browse** to locate the certificate.
3. Highlight the file and click **Open**.
4. Click **Next**.
5. Click **Done**.
6. Click **Save** on the Manage Trusted CAs screen.

To export a certificate:

1. Click **Export** under Action for the certificate you want to export.
2. On the Summary page, click the **Export** button.
3. Save the file on your system.

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 view certificate details:

- Click the certificate Serial number.

Manage SSL server certificates

PingFederate provides built-in SSL/TLS certificate management. Use this feature to establish and maintain the certificate(s) presented for access to the PingFederate administrative console and for incoming SSL/TLS connections at runtime (see *Set administration options* on page 130).

To create a new certificate:

1. Click **Create New**.
2. Enter the requested information on the form.
3. Click **Next**.
4. On the Summary screen, click **Done**.
5. Click **Save** on the Manage SSL Server Certificates screen.

Create certificate field descriptions

Field	Description
Common Name	The common name (CN) identifying 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.
Key Algorithm (drop-down menu)	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 (drop-down menu)	The signing algorithm of the certificate. (RSA-SHA256, SHA384 and SHA512; and ECDSA-SHA256, SHA384 and SHA512.)

To import a certificate and private key:

1. Click **Import**.
2. Click **Browse** to locate the certificate.
3. Highlight the file and click **Open**.
4. Enter the certificate password.
5. Click **Next**.
6. Click **Done**.
7. Click **Save** on the Manage SSL Server Certificates screen.

To view certificate information:

- Click its Serial number.

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

To activate a certificate:

1. Click **Activate for Runtime Server** or **Activate for Admin Console** 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.
2. Click **Save** on the Manage SSL Server Certificates screen.

To export a certificate:

1. Click **Export** under Action for the certificate you want to export.
2. Select **Certificate Only** on the Export Certificate screen.

   Or:

   Select **Certificate and Private Key** and enter an Encryption Password.
3. Click **Next**.
4. On the Certificate Summary screen, click **Export**.
5. Save the file on your system and click **Done**.

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 from the Main Menu.

   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. Select Generate Certificate Signing Request (CSR), if not already selected.
3. Click **Next**.
4. Click **Generate CSR** on the Generate CSR screen.
5. Click **Next**.
6. On the Certificate Summary screen, click **Export**.
7. Save the file on your system and click **Done**.

To import a certificate authority response:

1. Click **Certificate Signing** under Action for the relevant certificate.
2. Select Import CSR Response and click **Next**.
3. Click **Browse** and locate the CSR response to import.
4. Highlight the file and click **Open**.
5. Click **Next**.
6. Click **Done** on the Summary screen.
7. Click **Save** on the Manage SSL Server Certificates 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**.
2. Click **Save**.

Manage SSL client keys and certificates

You can create and manage your authentication private keys and the certificates your server presents as a client in an SSL/TLS transaction.

To create a new certificate:

1. Click **Create New**.
2. Enter the information on the form.
3. Click **Next**.
4. On the Summary screen, click **Done**.
5. Click **Save** on the Manage SSL Auth Private Keys and Certificates screen.

Create certificate field descriptions

Field	Description
Common Name	The common name (CN) identifying 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.
Key Algorithm (drop-down menu)	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 (drop-down menu)	The signing algorithm of the certificate. (RSA-SHA256, SHA384 and SHA512; and ECDSA-SHA256, SHA384 and SHA512.)

To import a certificate:

1. Click **Import**.
2. Click **Browse** to locate the certificate.
3. Highlight the file and click **Open**.
4. Enter the certificate password.
5. Click **Next**.
6. Click **Done** on the Import Certificate Details screen.
7. Click **Save** on the Manage SSL Auth Private Keys and Certificates screen.

To view certificate information:

- Click the certificate Serial number.

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

To export a certificate:

1. Click **Export** under Action for the certificate you want to export.
2. Select **Certificate Only**.

   Or:

Select **Certificate and Private Key** and enter an Encryption Password.

3. Click **Next**.
4. On the Certificate Summary screen, click **Export**.
5. Save the file on your system and click **Done**.

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 from the Main Menu.

   Note: 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. Select Generate Certificate Signing Request (CSR), if not already selected.
3. Click **Next**.
4. Click **Generate CSR** on the Generate CSR screen.
5. Click **Next**.
6. On the Certificate Summary screen, click **Export**.
7. Save the file on your system and click **Done**.

To import a certificate authority response:

1. Click **Certificate Signing** under Action for the relevant certificate.
2. Select Import CSR Response and click **Next**.
3. Click **Browse** and locate the CSR response to import.
4. Highlight the file and click **Open**.
5. Click **Next**.
6. Click **Done** on the Summary screen.
7. Click **Save** on the Manage SSL Auth Private Keys and Certificates screen.

To delete a certificate:

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

   Note: This option is inactive if the certificate is used in any connection or other type of configuration. To enable deletion, click **Check Usage** to locate the configuration(s) that depend on the certificate and then modify each to remove the dependency.
2. Click **Save**.

Manage digital signing and decryption keys and certificates

You can use PingFederate to create and maintain your server's signing certificates, which you may use to sign SAML requests, responses, and assertions. The same type of certificate is also used for XML decryption (see *XML encryption* on page 75).

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. Enter the information on the form.

3. Click **Next**.
4. On the Summary screen, click **Done**.
5. Click **Save**.

Create certificate field descriptions

Field	Description
Common Name	The common name (CN) identifying 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.
Key Algorithm (drop-down menu)	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 (drop-down menu)	The signing algorithm of the certificate. (RSA-SHA256, SHA384 and SHA512; and ECDSA-SHA256, SHA384 and SHA512.)

To import a certificate:

1. Click **Import**.
2. Click **Browse** to locate the certificate.
3. Highlight the file and click **Open**.
4. Enter the certificate password.
5. Click **Next**.
6. Click **Done**.
7. Click **Save**.

To view certificate information:

- Click the certificate Serial number.

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

To export a certificate:

1. Click **Export** under Action for the certificate you want to export.
2. Select **Certificate Only**.

   Or:

   Select **Certificate and Private Key** and enter an Encryption Password.
3. Click **Next**.
4. On the Certificate Summary screen, click **Export**.
5. Save the file on your system and click **Done**.

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 from the Main Menu.

   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. Select Generate Certificate Signing Request (CSR), if not already selected.
3. Click **Next**.
4. Click **Generate CSR** on the Generate CSR screen.
5. Click **Next**.
6. On the Certificate Summary screen, click **Export**.
7. Save the file on your system and click **Done**.

To import a certificate authority response:

1. Click **Certificate Signing** under Action for the relevant certificate.
2. Select Import CSR Response and click **Next**.
3. Click **Browse** and locate the CSR response to import.
4. Highlight the file and click **Open**.
5. Click **Next**.
6. Click **Done** on the Summary screen.
7. Click **Save** on the Manage Digital Signing Certificates screen.

To activate a certificate:

- Click **Activate** under Action for the certificate.

  This action is enabled only if more than one certificate is in the list.

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

Current certificate	The default value for the Creation Buffer field	The default value for the Activation Buffer field	The rotation window
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 (see *Configure runtime notifications* on page 130). 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 active 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 (see *Configure runtime notifications* on page 130). 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-Fedetion) 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://sso.example.com:9031/pf/federation_metadata.ping?**PartnerSpId**=SP1
An IdP partner with an entity ID of IdP1.	https://sso.example.com:9031/pf/federation_metadata.ping?**PartnerIdpId**=IdP1

Note that the base URL for the PingFederate runtime engine is https://sso.example.com:9031. For more information about the federation metadata endpoint, see *System-services endpoints* on page 452.

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 (see *Export metadata to an XML file* on page 101).

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 Connect to PingOne wizard in PingFederate 8.0 (or higher).

The certificate rotation settings are as follow:

Field	Values
Creation Buffer (days)	`90`
Activation Buffer (days)	`30`

Field	Values
Validity (days)	`1095`
Key Algorithm	`RSA`
Key Size	`2048`
Signature Algorithm	`SHA256withRSA`

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, PingFederate (the console node) uploads the new signing certificate to PingOne. 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 weeks ahead of expiry (see *Configure runtime notifications* on page 130).

Note: If you prefer to manually rotate the signing certificate, disable automatic certificate rotation. Note that PingFederate still automatically uploads your new signing certificate to PingOne to maintain SSO continuity through the managed SP connection.

For more information, see *Connect PingFederate to PingOne* on page 20 and *Connect to PingOne from Server Configuration* on page 142.

Manage certificate rotation settings

You manage certificate rotation settings for self-signed certificates in the **Server Configuration** > **Digital Signing & XML 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.

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 (see *Certificate validation* on page 73). Optionally, on the Manage Certificate Revocation screen you can enable and configure OCSP checking as the preferred verification method, depending on your requirements (see *OCSP Revocation Checking*).

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.

Also on the Manage Certificate Revocation screen, you can change system-default settings for CRL checking, as needed.

Note: No configuration changes are necessary on this screen if OCSP is not required for your federation deployment and the CRL defaults are acceptable.

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).

Field/Selection	Description
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.
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 160).
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.

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.incommonfederation.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. In the **Metadata URLs** screen, click **Add New URL**.
2. In the **URL** screen, enter the name and the URL of the metadata.
3. In 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. In 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. In 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. In 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

This portion of the Main Menu, under Server Configuration, allows 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 465).

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 42 in the "Supported Standards" chapter of the *Get started with PingFederate* guide and */sp/startAttributeQuery.ping* on page 448).

In addition, authentication is required to access PingFederate runtime data via JMX (see *Runtime monitoring using JMX* on page 131) 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 465).

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 validators

PingFederate® provides an authentication mechanism using plug-in password credential validators (PCVs). This feature provides centralized credential validation for other PingFederate configurations.

Multiple PCV instances are used by the HTML Form Adapter, the HTTP Basic Adapter, and the Username Token Processor. Multiple PCV instances can be applied to an IdP adapter or token processor. If an earlier PCV instance fails to authenticate the end user, control is returned to the IdP adapter or the token processor so that the next PCV instance can be tried.

Besides IdP adapters and token processors, PCV instances can also be mapped into the OAuth persistent grants in the **OAuth Settings** > **Resource Owner Credentials Mapping** screen for OAuth use cases using the password grant type.

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.

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**.

Choose a credential validator type

The first step in this configuration is choosing the type of the password credential validator (PCV). 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 *Ping Identity* (www.pingidentity.com/en/products/downloads/pingfederate.html).

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 credential validator instance

The configuration of a password credential validator varies depending on the credential validators deployed on your server. For credential validators bundled with PingFederate®, refer to one of the following topics:

- *Configure the LDAP credential validator* on page 174
- *Configure the simple credential validator* on page 176
- *Configure the RADIUS credential validator* on page 176
- *Configure the PingOne directory credential validator* on page 178

Configure the LDAP credential validator

The LDAP Username Password Credential Validator verifies credentials using an organization's LDAP data store.

In addition to look-up configuration, the **Instance Configuration** screen provides a means of defining error-message interpretation for LDAP stores other than Microsoft Active Directory (AD) or Oracle Directory Server (ODS). Message parsing for AD and ODS messages is built into PingFederate®; however, an administrator may also override the default message handling for those data stores on this screen.

Tip: LDAP server messages are used by the HTML Form Adapter to determine the LDAP password-change scenarios and to present the relevant messages to the end users.

1. Go to the **Server Configuration** > **Password Credential Validators** 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**.
2. On the **Type** screen:
   a) Enter a name and an ID for the password credential validator instance.
   b) Select the 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.
3. On the **Instance Configuration** screen, override authentication error messages as needed.

   Tip: This option may be required for an LDAP directory server other than 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** 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 for additional overrides as needed.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing value. Use the **Delete** and **Undelete** workflow to remove an existing value or cancel the removal request.

   Click **Move up** and **Move down** to change their display order. The ordering does not affect runtime processing.
4. Select the LDAP datastore 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 support the password change function in the HTML Form Adapter, ensure that the connection is secured using LDAPS; AD requires this level of security to allow password changes.
Search Base (Required)	The location in the LDAP directory server from which the search begins.
Search Filter (Required)	Query used to produce the desired set of matching records.

Field	Description
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** (the default) indicates a search of the base object and the entire subtree within the base object distinguished name.

Configure the simple 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.

1. Go to the **Server Configuration** > **Password Credential Validators** 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**.
2. On the **Type** screen:
   a) Enter a name and an ID for the password credential validator instance.
   b) Select the 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.
3. On the **Instance Configuration** screen, click **Add a new row to 'Users'**.
4. Enter the username and the password.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing value. Use the **Delete** and **Undelete** workflow to remove an existing value 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.
5. To complete the configuration:
   a) Click **Done** on the **Summary** screen.
   b) Click **Save** on the **Manage Credential Validator Instances** screen.

Configure the RADIUS 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.

1. Go to the **Server Configuration** > **Password Credential Validators** 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**.
2. On the **Type** screen:
   a) Enter a name and an ID for the password credential validator instance.

b) Select the 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.

3. On the **Instance Configuration** screen, 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.
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 value is **PAP**.
Shared Secret	The password shared between PingFederate and the RADIUS server used to encrypt passwords.

   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 for additional overrides as needed.

   Use the **Edit**, **Update**, and **Cancel** workflow to make or undo a change to an existing value. Use the **Delete** and **Undelete** workflow to remove an existing value 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.
4. 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.
Retry Count	The number of times to retry a failed connection before moving to the next host

Configure the PingOne directory credential validator

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

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

1. Go to the **Server Configuration** > **Password Credential Validators** 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**.
2. On the **Type** screen:
   a) Enter a name and an ID for the password credential validator instance.
   b) Select the 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.
3. On the **Instance Configuration** screen, 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	The REST API Client ID of your PingOne account, see *View or Renew Directory API Credentials* in the PingOne *Enterprise Administration Guide*.
Client Secret	The password for the REST API Client ID 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`.
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`.

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 credential validator 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 180).
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 179).

   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 180).

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 179).

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 fully qualified domain name of the PingFederate server

   `<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: `corporation.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 for the domain controller or KDC and click **Add**. For example: `fn_dc1` If unspecified, PingFederate uses a DNS lookup.
Test Domain/Realm Connectivity (button)	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.

Field	Description
	For multiple connections, note that the test stops at the first successful result, so all connections are not necessarily verified. Also, connectivity may be subsequently affected in different deployment scenarios, including for engine server nodes running in a clustered environment. **Tip:** For help resolving connection issues, select the **Debug Log Output** check box on the Manage Domain/Realm Settings screen, and run the test again to view the debug output to the PingFederate `server.log` (see *Manage domain connectivity settings* on page 181.

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 debugging output 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.

OAuth configuration

To use the OAuth Authorization Server (AS), start by enabling that role for PingFederate under **Server Settings** on the Roles and Protocols screen. Then configure the OAuth AS using options available under Server Configuration on the Main Menu, as described in this section.

For more information about the OAuth AS, see *PingFederate OAuth AS* on page 60.

Tip: Service providers may also add OAuth capabilities to the Browser SSO configuration for IdP connection partners, see *Configure OAuth attribute mapping*.

Enable the OAuth AS

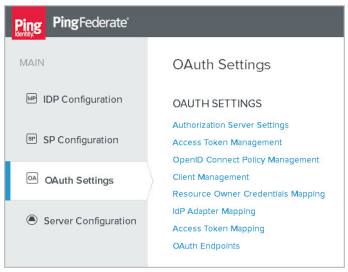
To configure OAuth use cases, you must enable the Authorization Server (AS) role in PingFederate®.

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 OAuth settings

Use the **OAuth Settings** menu to configure various settings to support the grant types that your applications require.



1. Configure the authorization server (AS) settings in the **OAuth Settings** > **Authorization Server Settings** screen.
2. Create one or more access token management instances in the **OAuth Settings** > **Access Token Management** screen.

   This is where you define the access-token's attribute contract for any given access token management instance.
3. Configure one or more mappings between the authentication sources and the persistent grant contract.

   Note that this is the first stage of the two-stage access token mapping process through the persistent grants.

   - For the authorization code or implicit grant types, configure a mapping in the **OAuth Settings** > **IdP Adapter Mapping** screen or the **IdP Connection** > **Browser SSO** > **OAuth Attribute Mapping** screen, or both.
   - For the resource owner grant type, configure mapping in the **OAuth Settings** > **Resource Owner Credential Mapping** screen.
4. 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 in the **OAuth Settings** > **Access Token Mapping** screen.

   Note that this is the second stage of the two-stage access token mapping process through the persistent grants.
5. For the SAML 2.0 profile for OAuth 2.0 authorization grants flow, configure a mapping in the **IdP Connection** > **OAuth SAML Grant Attribute Mapping** screen.

   Note that this is a direct mapping between the inbound assertions and the attribute contract of your access token management instances.
6. Define one or more OpenID Connect policies in the **OAuth Settings** > **OpenID Connect Policy Management** screen if you support OpenID Connect use cases and the **OpenID Connect** role has been enabled in the **Server Configuration** > **Server Settings** > **Roles and Protocols** screen.
7. Create one or more OAuth clients in the **OAuth Settings** > **Client Management** screen.
8. Optional: Configure client session management settings.

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 revocation Web Service to help OAuth Clients using the OpenID Connect protocol, 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 `/.well-known/openid-configuration`. 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.

To configure Asynchronous Front-Channel Logout:

1. Configure Asynchronous Front-Channel Logout settings in the **OAuth Settings** > **Authorization Server Settings** screen.
   a) Select the **Track User Sessions for Logout** check box to enable Asynchronous Front-Channel Logout for OpenID Connect clients
   b) Optional: Select the **Revoke User Session on Logout** check box 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.

Back-channel session revocation

Back-channel session revocation allows OpenID Connect 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 in the PingFederate administrative console. 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 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.

When a browser comes back to PingFederate with a revoked session, if the HTML Form Adapter is used, PingFederate assigns a new session identifier to the request and redirects the browser to the login form for the end user to sign on again. Other IdP adapters manage their own sessions; the end users may, or may not, need to re-authenticate.

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.

To 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 Settings** > **OpenID Connect Policy Management** screen.
3. Optional: In the **OAuth Settings** > **Authorization Server Settings** screen, modify the **Session Revocation Lifetime** value.

   Important: The **Session Revocation Lifetime** value should match or exceed the maximum session lifetime of the relying parties, including the **Session Timeout** value in all the applicable instances of the HTML Form Adapter.

   For example, the default **Session Revocation Lifetime** value of 250 minutes exceeds the default lifetime of the PingAccess (PA) Tokens by 10 minutes to allow for clock skew between servers (see *Web Sessions* in the PingAccess documentation).

Configure the AS settings

The **Authorization Server Settings** screen provides overall controls over the usage and behavior of the PingFederate® OAuth AS, including scope descriptions, authorization-code policy, and refresh-token and persistent-grant policy.

1. Go to the **OAuth Settings** > **Authorization Server Settings** screen.
2. Configure the PingFederate OAuth AS settings to suit your use cases.

   Refer to the following table for detailed information about each field.

Field	Description
Default Scope Description	Description of the permissions implied when no scope values are indicated or in addition to any values. This description displays when the user is prompted for authorization. **Tip:** If you want to localize the display text, you can enter a unique alias here, then use the same alias in language resource files.
Scope Value	A value that represents access to a resource or API on the Resource Server (RS). Applicable values require coordination with a developer or someone familiar with details of the RS OAuth implementation. **Tip:** For OpenID Connect, you can change the message presented to users for their authorization by entering `openid` under **Scope Value** and the message you want under **Scope Description**. **Important:** Also for OpenID Connect, to use the OpenID Connect defined scopes, you *must* add them under **Scope Value**, along with descriptions. • `profile` • `email` • `address` • `phone` **Note:** By default, scopes are available to all OAuth clients unless access is restricted on a per-client basis. **Caution:** Updating or removing a scope value in production will cause runtime errors if a client request specifies the scope using the previously defined value. In addition, errors will result if a requesting OAuth client is restricted to a scope whose value is no longer listed on the **Authorization Server Settings** screen, unless the affected client configuration is also updated.
Scope Description	A description of the scope value. This description appears when the user is prompted for authorization.

Field	Description
	Tip: If you want to localize the display text, you can enter a unique alias here, then use the same alias in language resource files.
Scope Group Value	A value that represents a group of scopes (a “super scope”) that you can reference in applicable OAuth 2.0 protocol interactions. The scopes within the scope group can be downgraded (upon refresh) into application-specific scopes with fewer permissions.
Scope Group Description	A description of the scope group value. This description appears when the user is prompted for authorization. **Tip:** If you want to localize the display text, you can enter a unique alias here, then use the same alias in language resource files.
Authorization Code Timeout (seconds)	The amount of time (in seconds) that an authorization code is considered valid.
Authorization Code Entropy (bytes)	The length (in bytes) of the authorization code returned to the OAuth client.
Persistent Grant Lifetime (blank for indefinite)	(Integer) Indicates the number of days or hours the OAuth AS stores persistent grants. Leave blank to maintain grants indefinitely. **Note:** You can also set persistent grant expiration for specific OAuth clients, overriding this global configuration.
Refresh Token Length (characters)	The number of characters the OAuth AS uses to generate the refresh token returned to the client.
Roll Refresh Token Values (default policy)	When selected, the OAuth AS generates a new refresh token value when a new access token is obtained. **Note:** New refresh token values are not issued during the interval defined by the **Minimum Interval to Roll Refresh Tokens** field. Not selecting the check box means the refresh token value is used until it becomes invalid (either by manually revoking or by some other security setting that renders it invalid).
Minimum Interval to Roll Refresh Tokens (hours)	The minimum number of hours that must pass before a new refresh token value can be issued. Provides a way to allow for rolling a refresh token value without having to send a new value on every request.
Reuse Existing Persistent Access Grants for Grant Types	If a client makes multiple requests for the same user and same (or lesser) scope, select the grant types you want the OAuth AS to reuse rather than creating a new grant for each request. Reusing an existing persistent grant imposes a limit of one grant per client. For example, in the case of refresh tokens, only one (the most recently issued) is valid, and the previously issued refresh token is invalidated. If multiple instances of the same client are used regularly by the same user, the associated check box should be cleared. When the **Implicit** grant type is selected, consent from the user is requested only for the first OAuth resource request associated with the grant. When the **Authorization Code** grant type 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, consent from the user is requested only once. The user is not asked for authorization on subsequent requests until the access grant is revoked. This

Field	Description
	function applies only when using the **Authorization Code** grant type *and* when the **Reuse Existing Persistent Access Grants for Grant Types** check box is selected. **Tip:** On a per-client basis, user consent may be bypassed completely, overriding this setting.
Allow Unidentified Clients to Request Resource Owner Password Credentials Grants	When selected, allows Resource Owners to obtain access tokens without defining a client.

Allow Unidentified Clients to Request Extension Grants	When selected, allows user-initiated or client-initiated events (for example, a mobile application or scheduled task) to obtain access tokens without the client presenting a `client_id` or `client_secret` for the extension grant types, namely:
	• Validation grant type
	`urn:pingidentity.com:oauth2:grant_type:validate_bearer`
	• SAML 2.0 Bearer Assertion grant type
	`urn:ietf:params:oauth:grant-type:saml2-bearer`

Persistent Grant Extended Attributes	Extend persistent grants to include additional attributes from your authentication systems.
Password Credential Validator	Required for HTTP Basic authentication if the OAuth REST Web Service is used for managing client applications, the OAuth Access Grant Management Service is used for managing access grants, or both. These service cannot be used if a validator is not provided.
The following settings require the selection of the **OpenID Connect** role in the **Server Configuration** > **Server Settings** > **Roles & Protocol** screen.	
Track User Sessions for Logout	When selected, PingFederate links the IdP adapter sessions that are used by OAuth clients and SLO-enabled (SAML 2.0 and WS-Federation) connections for end users. When an end user logs out of one session, PingFederate sends (via the browser) logout requests to close other associated sessions.
Revoke User Session on Logout	When selected, PingFederate revokes the corresponding sessions on logout. **Note:** PingFederate always adds the session to the revocation list, even if a logout error occurs.
Session Revocation Lifetime (minutes)	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.

OAuth access token management

PingFederate® supports multiple access token management instances. This capability allows you (the administrator) 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 a certain RS.

Clients can specify an access token management instance by providing the access token manager ID (`access_token_manager_id`) or a resource URI (`aud`) in their requests to the PingFederate OAuth AS at the token endpoint (`/as/token.oauth2`) and the authorization endpoint (`/as/token.oauth2`).

When defining an access token management instance, you can customize various settings, including the token format, lifetime and attribute contract for this instance. You can also limit the access token management 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 access token management instance.

You can also add the desired RS clients to the ACL, such that these are RS clients can ensure that the access token they have received was issued by a specific access token management instance. They do this by specifying which instance will be used in validating the token (via the `access_token_manager_id` or `aud` parameter) they want to use in the token validation request.

At runtime, the OAuth AS uses the following rules to determine which access token management instances it should use:

1. Limit the eligible access token management 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 SAML Grant requests, it is the set of instances for which a mapping is defined in the IdP connection. Furthermore, the client ACL (if configured) can also limits which access token management instances are eligible.
2. If the request comes with an `access_token_manager_id` or `aud`, the OAuth AS uses the information to determine the applicable access token management instance.
3. If the request does not come with either parameter, for OpenID Connect clients (where the scope value contains `openid` in their requests), the OAuth AS uses the access token manager specified by the OpenID Connect policy of the client.
4. If the request comes with neither of the two parameters nor the `openid` scope, the OAuth AS uses the default access token manager of the client (if configured) or the default instance defined for the installation.

If no match can be found in the eligible list, the OAuth AS aborts the request.

Manage access token management instances

Use the **OAuth Settings** > **Access Token Management** workflow to specify how the PingFederate® OAuth AS manages access tokens.

To configure a new instance:

- Click **Create New Instance**.

To edit an existing access token management instance:

- Click the instance name and click the step you need to change.

To delete an access token management instance:

1. Click **Delete** next to the instance name. (To undo the deletion, click **Undelete**.)

   Note: This option is inactive if the instance is referenced elsewhere, or if it is a parent to other instances. To enable deletion, click **Check Usage** to locate the configurations that depend on the instance and go to each listed configuration to remove the dependencies. If the instance you want to delete is a parent, delete child instances first.
2. Click **Save** to confirm the deletion.

Define an access token management instance

Use the **Type** screen to start creating a new or editing a current access token management instance.

To create a new instance:

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 *Support Center* (pingidentity.com/support)
3. Optional: Select a **Parent Instance** from the list.

If you are creating an instance that is similar to an existing instance, you might consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

4. Click **Next**.

To edit an existing access token management instance:

1. Edit the instance name as needed.
2. 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.
3. Click **Next**.

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 188
- *Configure JSON-token management* on page 189

Configure reference-token management

1. Modify the default values as needed.

   Refer to the following table for detailed information about each field.

Field	Description
Token Length	The number of characters the AS uses to define the token reference. Increasing the length will enhance token security, if desired. (Maximum: 256)
Token Lifetime	The amount of time (in minutes) that an access token is considered valid.
Lifetime Extension Policy	Indicates whether the OAuth AS should reset token lifetimes each time a token is validated. The token plug-in checks the policy before updating the lifetime of an access token. Options are: no extension policy, reset token lifetimes only for transient tokens (not backed by a persistent policy), or reset lifetimes for all tokens.
Maximum Token Lifetime (Optional)	Defines an absolute maximum token lifetime (in minutes) for use with the Lifetime Extension Policy. An access token's lifetime can be extended but not beyond this setting. You must specify a value that is greater than or equal to the value specified in the Token Lifetime.
Lifetime Extension Threshold Percentage	When PingFederate® is deployed in a cluster and token-lifetime extension is enabled, there must be a cluster-group remote procedure call (RPC) to extend the life of a token. This setting limits RPC overhead by suspending the calls until the set threshold is crossed. For example, if the token lifetime is one hour and the threshold is 50%, the lifetime will not be extended until the remaining time is less than 30 minutes. This option could potentially reduce RPC traffic between nodes by orders of magnitude while still supporting the LifeTime Extension Policy.
Advanced Fields	
Mode for Synchronous RPC	Some RPC events require that the caller get some data from the remote nodes, so the call is synchronous and blocks waiting on the responses. This configuration setting indicates whether the caller should wait for a response from all nodes in the cluster

Field	Description
	or just a majority of nodes. This is designed to eliminate the need for a complete state synchronization at startup. Synchronous RPC calls occur when a node receives a verification request for a token it does not recognize and for token issuance.
RPC Timeout	Timeout between cluster nodes during synchronous communication. Recommended setting is from 100 milliseconds to 1000 (1 second).

2. Click **Next**.

Configure JSON-token management

You can configure settings for bearer JSON Web Tokens (JWTs)— secure, self-contained tokens that can be validated locally by the target RS.

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.

1. Add one or more symmetric keys, signing certificates, or both.

   Click the **Add a new row . . .** links, enter information, and then click **Update** under **Action**.

   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: Signing certificate key length must be at least 2,048 bits.

2. Change or select entries for required fields and make other changes as needed.

   Refer to the following table for detailed information about each field.

Field	Description
Token Lifetime (Required)	The amount of time, in minutes, that an access token is considered valid.
JWS Algorithm (Required)	The hash-based message authentication code (HMAC) or RSA signing algorithm used to protect the integrity of the token.
Active Symmetric Key ID	The Key ID of the key to use when producing JWTs using an HMAC-based algorithm. Required if an HMAC-based JWS algorithm is selected from the **JWS Algorithm** list.
Include Key ID Header Parameter	When selected (the default), the key ID is used in the `kid` header parameter for the token.
Active Signing Certificate Key ID	The Key ID of the key pair and certificate to use when producing JWTs using an RSA-based algorithm. Required if an RSA-based JWS algorithm is selected from the **JWS Algorithm** list.
Include X.509 Thumbprint Header Parameter	When selected, the X.509 Certificate Thumbprint is used in the `x5t` header parameter for the token.
Client ID Claim Name	The name of a JWT claim used to represent the OAuth Client ID.
Scope Claim Name	The name of a JWT claim used to represent the scope of the grant.
Advanced Fields	
Space Delimit Scope Values	When selected, indicates scope strings will be delimited by spaces rather than represented as a JSON array (the default).
Issuer Claim Value	The value of the Issuer (`iss`) claim in the JWT.

Field	Description
Audience Claim Value	The value of the Audience (`aud`) claim in the JWT.
JWT ID Claim Length	Indicates the number of characters of the JWT ID (`jti`) claim in the JWT. (If `0`, no claim is included.)
Access Grant GUID Claim Name	The name of the JWT claim used to carry the persistent access grant GUID. If the claim is present during validation, the grant database is consulted to ensure the grant is still in force.
Publish Key ID X.509 URL	Indicates whether certificates will be made accessible by Key ID at `https://<pf_host>:<port>/ext/oauth/x509/kid?v=<id>`
Publish Thumbprint X.509 URL	Indicates whether certificates will be made accessible by thumbprint at `https://<pf_host>:<port>/ext/oauth/x509/x5t?v=<base64url encoded SHA-1 thumbprint>`

3. Click **Next**.

Define the access token attribute contract

On the **Access Token Attribute Contract** screen, create a list of attributes to be referenced in an OAuth access token. You must enter at least one attribute.

To add an attribute:

1. Enter the attribute name in the text box.
2. Click **Add**.

   Add additional attributes as needed. Attribute names are case-sensitive and must correspond to the attribute names expected by the Resource Server.

To modify an attribute name:

1. Click **Edit** under Action for the attribute.
2. Make the change and click **Update**.

To delete an attribute:

- Click **Delete** under Action for the attribute.

Click **Next** when finished editing attributes.

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.

To add a resource URI:

1. Enter the value in the text box.
2. Click **Add**.

   Enter additional resource URIs as needed. The resource URIs must correspond to the resource expected by the Resource Server.

To modify a resource URI:

1. Click **Edit** under Action for the resource URI.
2. Make the change and click **Update**.

To delete a resource URI:

- Click **Delete** under Action for the resource URI.

Click **Next** when finished editing resource URIs. (To undo the deletion, click **Undelete**.)

Define access control

On the **Access Control** screen, you may restrict which OAuth clients are allowed to use this access token management instance.

To enable access control:

1. Select the Restrict Allowed Clients check box.
2. Select a client under Allowed Clients and click **Add**.
3. Select additional clients as needed.

To remove a previously allowed client:

- Click **Delete** next to the applicable client. (To undo the deletion, click **Undelete**.)

To disable access control:

- Clear the Restrict Allowed Clients check box.

Click **Next** when finished configuring access control.

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**.

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 begin creating a new policy:

1. Click **Add Policy**.
2. On the **Manage Policy** screen, enter the required information and configure optional settings.
   a) Enter a policy ID and name.
   b) Select an access token manager instance from the list
   c) Change the default value of the ID token lifetime in minutes.

      The default value is 5 and can range from 0 to 65,535.
   d) Select the **Include Session Identifier in ID Token** check box to add a session identifier in the ID tokens.
   e) 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 (`/idp/userinfo.openid`).
   f) Click **Next**.

To edit an existing policy:

- Click its link and then on the **Summary** screen click the heading over the information that needs updating.

Configure the policy attribute contract

On the **Attribute Contract** screen, specify attributes you want returned from the PingFederate® UserInfo endpoint for OpenID Connect (`/idp/userinfo.openid`).

Note: The `sub` value of this configuration is also used to fulfill an internally managed ID Token attribute contract. The contract usage is dynamic, depending on how and when the user authenticated.

To add an attribute:

- Enter the attribute name in the text box and click **Add**.

To modify an attribute name:

1. Click **Edit** under Action for the attribute.
2. Make the change and click **Update**.

To delete an attribute:

- Click **Delete** under Action for the attribute.

Configure policy attribute sources and user lookup

Attribute sources are specific data store or directory locations containing information that may be needed to fulfill the attribute contract for a policy. This configuration is optional.

1. Click **Add Attribute Source** (or click **Next** if you are not using a data store).
2. On the **Data Store** screen, enter values into the **Attribute Source ID** and **Attribute Source Description** fields, select an data store from the list, and click **Next** to continue the configuration.

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.

  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** under **Source** and **Authenticating Authority** under **Value**. 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.

  Note: The HTTP Request 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. (If the Expression selection is not listed, then the feature is not enabled.

- 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 (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 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. Choose a source and then choose (or enter) a value for each attribute in the contract.
2. Click **Next**.

Specify issuance criteria for policy mapping

Use the **Issuance Criteria** optional screen to define criteria PingFederate® can evaluate to determine whether to issue the ID token and access token. If a criterion fails, no tokens are issued and access is denied.

To configure issuance criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.
   - Mapped Attributes – Select to access the output data returned from UserInfo endpoint.
   - Access Token – Select to use attributes from the access token.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.
8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations (for example, OR conditions) using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

- Click **Add**. Continue to add expressions, as needed.
- Click the **Test** link to input values and test the resulting output for the expression.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Policies screen).

To edit issuance criteria:

- Click **Edit** under Actions for the criteria.

To delete issuance criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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.

Manage OAuth clients

An OAuth client application interacts with the PingFederate® OAuth AS to obtain access tokens needed to call OAuth-protected services at the Resource Server (RS).

Important: OAuth client records, like other PingFederate configuration data, are maintained in XML files. Alternatively, you can define an OAuth client data store (a centralized database) to store client records.

- To add a client, click **Add Client** and complete the configuration on the **Clients** screen.
- To edit a client, click the client ID and make the necessary changes on the **Clients** screen.

  Tip: Use the **Search** features to locate clients that may be out of view or on subsequent pages, or use the page-navigation controls, when present. Click **Clear** to redisplay the list after a search. (Select Advanced Search for search options.)
- To remove a client or cancel the removal request, use the **Delete** and **Undelete** workflow next to the applicable client, and then click **Save**.

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.

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 to identify itself. This identifier is included with every request the client makes.
Client Authentication	A selection other than **None** is required only for the **Client Credentials** grant type. When required, select *either*: • **Client Secret** for HTTP Basic authentication. Click **Generate Secret** to create a strong random alphanumeric string or manually enter a secret. If this button is disabled, select the **Change Secret** check box. *Or*:

Field	Description
	• **Client Certificate** for mutual SSL/TLS authentication—recommended for application configurations where security policies prohibit storing passwords. **Note:** Mutual SSL/TLS authentication requires the use of a secondary PingFederate SSL port. Enter information into the following required fields: **Client TLS Certificate Issuer** – Select a Trusted CA from the list. Alternatively, you may choose to trust any of the issuers listed in the list. (These are CA certificates imported into PingFederate.) **Client TLS Certificate Subject DN** – Enter the client-certificate subject DN, or click **Browse** to locate the certificate and then **Extract** to retrieve the DN.
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.
Redirection URIs	URIs to which the OAuth AS may redirect the resource owner's user agent after authorization is obtained. A redirection URI is used with the **Authorization Code** and **Implicit** grant types. Enter a fully qualified URL and click **Add** for each entry required. Wildcards are allowed. However, for security reasons, make the URL as restrictive as possible. For example: `https://www.company.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 Scopes	Restricts this client's access to specific scopes or scope groups. When selected, a list appears of all scopes defined within the PingFederate AS. Select the scopes or scope groups available for this client. **Note:** Selecting this box and not selecting any specific scopes restricts this client's access to only the default scope defined in the **OAuth Settings** >

Field	Description
	Authorization Server Settings screen. If a client requests a specific scope (not selected in the list), an error is returned.
Allowed Grant Types	Available grant types that this client is allowed to use.
Default Access Token Manager (optional)	Select a default access token manager instance for this client.
Persistent Grants Expiration	Allows an administrator to override the **Persistent Grant Lifetime** field value set globally in the **OAuth Settings** > **Authorization Server Settings** screen.
Refresh Token Rolling Policy	Select **Don't Roll** or **Roll** to override the **Roll Refresh Token Values** setting defined in the **OAuth Settings** > **Authorization Server Settings** screen. Leave **Use Global Setting** selected to default to the **Roll Refresh Token Values** setting defined in the **OAuth Settings** > **Authorization Server Settings** screen.
OpenID Connect	**Note:** These options are displayed only when the **OpenID Connect** role is enabled in **Server Configuration** > **Server Settings** > **Roles & Protocols** screen. **ID Token Signing Algorithm** (Optional): Select the signing algorithm for the ID Token. The default algorithm is **RSA using SHA-256**. **Policy** (Optional): Choose a specific OpenID Connect policy. **Grant Access to Session Revocation API**: When selected, this client application is allowed 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. **Note:** If the **Track User Sessions for Logout** check box is selected in the **OAuth Settings** > **Authorization Server Settings** screen, there are two additional settings: **PingAccess Logout Capable** (Optional): 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). **Logout URIs** (Optional): Enter additional endpoints at the relying parties as needed. PingFederate sends (via the browser) requests to these URIs as part of the logout process.

Manage resource-owner credentials mappings

The **OAuth Settings** > **Resource Owner Credentials Mapping** workflow allows you to map attributes based on validated user credentials into the `USER_KEY` and extended attributes for a persistent grant. You may supplement credential-validation mapping sources with attribute look-ups from your user data source.

To create a mapping:

- Select an password credential validator instance from the **Source Password Validator Instance** list and then click **Add Mapping**.

  If the list does not contain any instances or the instance you want, use the **Server Configuration** > **Password Credential Validators** workflow to define one.

To edit an existing mapping:

- Click its link and then on the **Summary** screen click the heading over the information that needs updating.

Configure resource-owner attribute sources and user lookup

Attribute sources are specific data store or directory locations containing information that may be needed to fulfill the `USER_KEY` and any extended attribute values of persistent grants. This configuration is optional.

1. Click **Add Attribute Source** (or click **Next** if you are not using a data store).
2. On the **Data Store** screen, enter values into the **Attribute Source ID** and **Attribute Source Description** fields, select an data store from the list, and click **Next** to continue the configuration.

Define resource-owner contract fulfillment

On the **Contract Fulfillment**screen, you map values into the `USER_KEY` and extended attributes for a persistent grant. Use this mapping for the resource owner password credential grant type.

Note: The `USER_KEY` values must be unique across all end users because it is the identifier to store and to retrieve persistent grants. For example, if you have two Active Directory domains, the sAMAccountName 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 Subject DN to the `USER_KEY`.

Map each attribute from one of these Sources:

- Context

  Values are returned from the context of the transaction at runtime.

  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** under **Source** and **Authenticating Authority** under **Value**. 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.

  Note: The HTTP Request 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. (If the Expression selection is not listed, then the feature is not enabled.

- 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 (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 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**.

Specify issuance criteria for resource-owner credentials mapping

Use the **Issuance Criteria** screen to 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.

To configure issuance criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.
   - Mapped Attributes – Select to access the output of the mapping into the persistent grant—for example, the USER_KEY or an extended attribute.
   - Password Credential Validator – Select to access attributes returned from the credential-validation instance.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.
8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations (for example, OR conditions) using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.
9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the **Manage Mappings** screen).

To edit issuance criteria:

- Click **Edit** under Actions for the criteria.

To delete issuance criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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.

Manage IdP adapter mappings for OAuth

The **OAuth Settings** > **IdP Adapter Mapping** workflow allows you to map attributes based on an IdP adapter configuration into the `USER_KEY` and extended attributes for a persistent grant, and the `USER_NAME` (presented to the user for authorization permission). You may supplement values returned from the adapter with attribute look-ups from your user data source.

Use this mapping for the authorization code and implicit grant types.

To create a mapping:

- Select an IdP adapter instance from the **Source Adapter Instance** list and then click **Add Mapping**.

To edit an existing mapping:

- Click its link and then on the **Summary** screen click the heading over the information that needs updating.

Configure IdP adapter attribute sources and user lookup

Attribute sources are specific data store or directory locations containing information that may be needed to fulfill the `USER_KEY` and any extended attributes of the persistent grants. In addition, attribute sources can be used to fulfill the `USER_NAME` presented to the end user for authorization permission. This configuration is optional.

1. Click **Add Attribute Source** (or click **Next** if you are not using a data store).
2. On the **Data Store** screen, enter values into the **Attribute Source ID** and **Attribute Source Description** fields, select an data store from the list, and click **Next** to continue the configuration.

Define grant contract fulfillment for IdP adapter mapping

On the **Contract Fulfillment**screen, you map values into the `USER_KEY` and extended attributes for a persistent grant, and the `USER_NAME` for the user's display name on the authorization page.

Note: The `USER_KEY` values must be unique across all end users because it is the identifier to store and to retrieve persistent grants. For example, if you have two Active Directory domains, the sAMAccountName 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 Subject DN to the `USER_KEY`.

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.

  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** under **Source** and **Authenticating Authority** under **Value**. 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.

Note: The HTTP Request 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. (If the Expression selection is not listed, then the feature is not enabled.

- 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 (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 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**.

Specify issuance criteria for OAuth IdP adapter mapping

Use the **Issuance Criteria** screen to 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.

To configure issuance criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Adapter – Select to access attributes from the IdP Adapter.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.
   - Mapped Attributes – Select to access the output of the mapping into the persistent grant—for example, the USER_KEY or an extended attribute.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).

5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.
8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.
   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.
9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the **Manage Mappings** screen).

To edit issuance criteria:

- Click **Edit** under Actions for the criteria.

To delete issuance criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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.

Manage access token mappings

In this required configuration, an administrator maps attributes to be requested from the OAuth resource server with the access token—the token attribute contract.

When mapping a Default context, you define how the OAuth AS maps values into the attributes based on the persistent-grant `USER_KEY` and any extended attributes.

When a specific context is selected, you can also map attributes from the selected context, namely the chosen IdP adapter, credential validator, or IdP connection (with an OAuth attribute mapping configuration) into the access tokens.

The mapping used at runtime depends on the authentication context of the original grant. If the authentication context results a match, the OAuth AS uses that specific mapping; otherwise, it uses the default mapping for the applicable access token manager instance.

Note: The **OAuth Settings** > **Access Token Mapping** workflow becomes available only after at least one instance of access token management has been configured.

To create a token mapping:

1. Select a context from the list.

   Contexts for mappings include:

   - The default mapping from the persistent grants when no other contexts are configured
   - The mappings from the IdP adapter instances (based on the **IdP Adapter Mapping** configuration) and the persistent grants
   - The mappings from the password credential validator instances (based on the **Resource Owner Credentials Mapping** configuration) and the persistent grants
   - The mappings from the IdP connections (based on the **OAuth Attribute Mapping** configuration) and the persistent grants
2. Select an access token manager instance from the list.
3. Click **Add Mapping**.

To edit an existing mapping:

- Click its link and then on the **Summary** screen click the heading over the information that needs updating.

Configure access token attribute sources and user lookup

Attribute sources are specific data store or directory locations containing information that may be needed to fulfill the token attribute contract. This configuration is optional.

1. Click **Add Attribute Source** (or click **Next** if you are not using a data store).
2. On the **Data Store** screen, enter values into the **Attribute Source ID** and **Attribute Source Description** fields, select an data store from the list, and click **Next** to continue the configuration.

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/IdP Connection

  If you selected a specific IdP adapter, password credential validator or IdP connection under Context in the previous screen, you have the option to map attributes from that specific authentication system. Select the corresponding Context in this screen and choose the desired attribute under Value.
- Context

  Values are returned from the context of the transaction at runtime.

  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** under **Source** and **Authenticating Authority** under **Value**. 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.

  Note: The HTTP Request 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. (If the Expression selection is not listed, then the feature is not enabled.
- Persistent Grant

  When you make this selection, the associated Value drop-down list is populated by the `USER_KEY` and extended attributes from the persistent access-token grant.
- LDAP/JDBC/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 (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 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**.

Specify issuance criteria for access token mapping

Use the **Issuance Criteria** screen to 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 example, before reissuing a token in a refresh-token scenario, the server can verify that the user is still current in your organization's user-data store and has retained the necessary permissions.

To configure issuance criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.
   - Mapped Attributes –Select to use access token attributes returned from token attribute contract mapping.
   - Persistent Grant – When you select this option, the associated Attribute Name drop-down list is populated by the `USER_KEY` and extended attributes from the persistent access-token grant.

2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).

5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.
9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the **Manage Mappings** screen).

To edit issuance criteria:

- Click **Edit** under Actions for the criteria.

To delete issuance criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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 SAML Grant IdP connection

An OAuth SAML Grant connection exchanges a SAML assertion for an OAuth access token with the PingFederate® OAuth AS. You can configure an OAuth SAML Grant connection with an IdP partner either in conjunction with browser-based SSO, WS-Trust, or independently.

1. On the **Connection Type** screen, select the **OAuth SAML 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 SAML Grant** configuration.
2. On the **General Info** screen, enter the required information.
3. On the **OAuth SAML Grant Attribute Mapping** screen, click **Configure OAuth SAML Grant Attribute Mapping** to continue.

Define an attribute contract for the OAuth SAML Grant

An attribute contract is a set of user attributes the IdP sends in the SAML assertion for this connection. You identity these attributes on this screen.

SAML_SUBJECT is always sent in a SAML assertion and contains the name identifier of the user for whom the access token is being requested.

Optionally, you can mask the values of attributes (other than SAML_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 SAML Grant attribute mapping

This optional configuration is the same for all OAuth attribute-mapping task flows.

- If you do not need additional attributes from a data store, just click **Next** on the **Data Store** screen.

Define OAuth SAML Grant contract fulfillment

The last step in configuring OAuth SAML Grant attribute mapping is to map SAML grant attributes to the attribute values of the access token.

Map attributes from one of the following Sources:

- Assertion

Values are contained in the assertions from this IdP. When you make this selection, the associated Value drop-down list is populated by the attribute contract.

- Context

  Values are returned from the context of the transaction at runtime.

  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** under **Source** and **Authenticating Authority** under **Value**. 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.

  Note: The HTTP Request 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. (If the Expression selection is not listed, then the feature is not enabled.

- JDBC/LDAP/Custom

  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 (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 assertion, 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 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**.

Specify issuance criteria for OAuth SAML Grant

Use the **Issuance Criteria** screen to 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.

To configure issuance criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Assertion – Select to access attributes from the assertion.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.
   - Mapped Attributes – Select to access the output of the mapping into the access token.
2. Select an attribute name.
3. Select the Condition you want to apply.

Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).

5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the **Manage Mappings** screen).

To edit issuance criteria:

- Click **Edit** under Actions for the criteria.

To delete issuance criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

Review OAuth SAML Grant attribute mapping

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 SAML 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**.

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 191
- *Manage resource-owner credentials mappings* on page 196
- *Manage IdP adapter mappings for OAuth* on page 199
- *Manage access token mappings* on page 201

A similar configuration is also used for attribute mapping in an IdP connection (see *Configure OAuth attribute mapping* on page 343) and when configuring an OAuth SAML Grant connection (see *Configure an OAuth SAML Grant IdP connection* on page 204).

(For additional information, see *Fulfillment by data store queries* on page 494.)

Choose a data store

From the **Data Store** screen, choose a data store for PingFederate to look up attributes.

1. Enter an ID and a description 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 496 • *Enter a database search filter* on page 496
LDAP	• *Specify an LDAP directory and attributes* on page 497 • *Define encoding for LDAP binary attributes* on page 498 (optional) • *Enter an LDAP search filter* on page 499
Custom	• *Specify a custom source filter and fields* on page 499

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 and mapping such policy contract to multiple connections.

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 examples, 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 selectors

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**.

Choose a selector type

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

The configuration of an authentication selector 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 209
- *Configure the Cluster Node Authentication Selector* on page 210
- *Configure the Connection Set Authentication Selector* on page 211
- *Configure the HTTP Header Authentication Selector* on page 211
- *Configure the HTTP Request Parameter Authentication Selector* on page 212
- *Configure the OAuth Scope Authentication Selector* on page 213
- *Configure the Requested AuthN Context Authentication Selector* on page 214

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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.
2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.
3. On the **Authentication Selector** screen:
   a) Click **Add a new row to 'Networks'**, enter a network range, and then click **Update**.
   b) Repeat the previous step as needed for additional 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.

c) 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.

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`.

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 another selector 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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.

2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.
3. On the **Authentication Selector** screen, click **Next** to proceed to the next screen.
4. On the **Selector Result Values** screen, list the index numbers for each of the cluster nodes.

   Repeat this step to specify additional entries as needed.

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

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

a) Click **Done** on the **Summary** screen.
b) Click **Save** on the **Manage Authentication Selector Instances** screen.

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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.
2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.
3. On the **Authentication Selector** screen:
   a) Click **Add a new row to 'Connections'** under **Action**.
   b) 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 configured here. If a match is found, the selector returns a result value of `Yes`.
   c) Repeat the previous step as needed for additional connections.

   There is no priority given to the order in which the connections appear.

   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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.

2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.
3. On the **Authentication Selector** screen:
   a) Click **Add a new row to 'Results'** under **Action**.
   b) Enter an expression for use when inspecting the HTTP header value of the target HTTP header and click **Update**.

      This field is case-sensitive, and wildcard entries are allowed; for example, `*Firefox*`.

      The following expressions work to identify these commonly used browsers.

Browser	Expression
Chrome	`*Chrome*`
Firefox	`*Firefox*`
iPhone	`*iPhone*`
iPad	`*iPad*`
Internet Explorer	`*MSIE*`
Safari	`*Safari*`

      For information about Internet Explorer 11 (or higher), see *User-agent string changes* from Microsoft (msdn.microsoft.com/library/hh869301.aspx).

      At runtime, the selector compares the HTTP header to these wildcard expressions. If a match is found, the selector returns a result value of `Yes`.
   c) Repeat the previous step as needed to add more wildcard expressions.

      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.
   d) 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.
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 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 HTML Form Adapter or between a Kerberos Adapter and an X.509 Adapter. 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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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**.

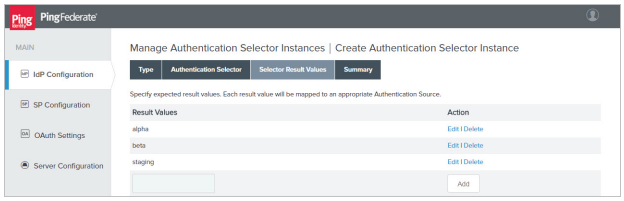
   Note: You can only remove a selector instance if it is not deployed in any authentication policy.
2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the type from the list.
   c) Optional: Select a **Parent Instance** from the list.

   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.
3. On the **Authentication Selector** screen, enter the exact parameter name in the **HTTP Request Parameter Name** field. This field is case-sensitive.
4. On the **Selector Result Values** screen, enter the exact parameter values.

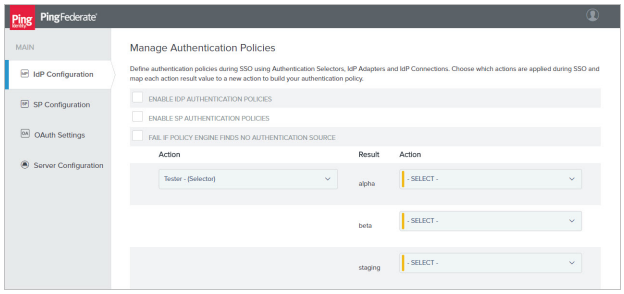
   Parameter values are case-sensitive.

   Repeat this step to specify additional entries as needed.

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



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



5. 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 (OAuth 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.

1. Configure one or more scopes in **OAuth Settings** > **Authorization Server Settings** screen if you have not already done so.
2. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.
3. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.
4. On the **Authentication Selector** screen, select the required scopes.

   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.
5. 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. Go to the **Selectors** screen from the **IdP Configuration** or **SP Configuration** menu.
   - 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 only remove a selector instance if it is not deployed in any authentication policy.
2. On the **Type** screen:
   a) Enter a name and an ID for the authentication selector.
   b) Select the 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.

3. On the **Authentication Selector** screen, select the **Add or Update AuthN Context Attribute** check box if you want to update the AuthN Context attribute value with the value specified in the **Selector Result Values** screen accordingly.

   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

4. 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) 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.

   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.

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

Policies

An authentication policy is a tree of authentication sources, selector instances, and authentication policy contracts, on which the decisions to route the request to a series of approved authentication sources with an optional authentication policy contract at the end or deny the request are based. Administrators can create one or more authentication policies to fulfill their authentication requirements.

IdP authentication policies apply to IdP Browser SSO requests, adapter-to-adapter requests, and browser-based OAuth authorization-code and implicit flows. SP authentication policies on the other hand apply to SP-initiated Browser SSO requests received by the PingFederate SP server at the `/sp/startSSO.ping` endpoint.

The order of authentication policies matters because the policy engine in PingFederate evaluates policies from top to bottom.

At runtime, the policy engine derives an authentication tree from the applicable policies and either approves or denies a request.

Policy, paths, and authentication policy contracts

An authentication policy starts with either a selector or an authentication source. Authentication sources and most selectors have two results (**Success** or **Fail**, **Yes** or **No**). Each result forms a path.

A 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 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.

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 policy contract for different closed-ended paths (in one or more authentication policies) and fulfill them differently to suit the requirements. For IdP Browser SSO use cases, the same policy contract can also be mapped to multiple SP connections, thus enforcing the same set of authentication policies to those connections.

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 any closed-ended paths that are not applicable to the request, which may skip one or more policies if necessary. A path is considered not applicable to a request if the authentication policy contract at the end of the path (or the last authentication source in such path when it does not end with an authentication policy contract) is not mapped to the corresponding use case or is blocked by the virtual server ID included in the request (see *Working with multiple virtual server IDs*). Note that virtual server IDs are not applicable to adapter-to-adapter mappings or OAuth uses cases.

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 that 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.

Define authentication policies

You manage authentication policies and settings in the **Policies** > **Manage Authentication Policies** screen.

1. Select the **Enable IdP Authentication Policies** check box to turn on authentication policies, which apply to 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 selected in the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

   If you are still configuring policies and connections, clear this check box for now. When you are ready, select the option to activate the policies.
2. Select the **Enable SP Authentication Policies** check box to turn on authentication policies, which apply to SP-initiated Browser SSO requests received by the PingFederate SP server at the `/sp/startSSO.ping` endpoint.

   This check box is only visible when the SP role is selected in the **Server Configuration** > **Server Settings** > **Roles & Protocols** screen.

   If you are still configuring policies and connections, clear this check box for now. When you are ready, select the option to activate the policies.
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.
4. Create an authentication policy.

   Tip: Think of authentication sources and selectors as checkpoints when implementing your authentication requirements.

   a) Select an authentication source (an IdP adapter or an IdP connection) or a selector from the **Action** list.

      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.

      For the PingID™ Adapter, IdP adapters developed using the `IdpAuthenticationAdapterV2` interface from the PingFederate SDK (including the bundled 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.

      All outcomes (including those based on rules) are displayed under **Result**. Each result forms a path.

   b) For each path, select an action from the list.

      - If additional processing is required, repeat *step 4a*.
      - If the 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.

        A 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.

        If you need to reuse an authentication policy in multiple connections, select an authentication policy contract as the last action of a path, configure its contract fulfillment, and map the authentication policy contract to the applicable connections. Click **Contract Mapping** and follow the on-screen instructions to configure the fulfillment of the authentication policy contract.

        If the authentication policy contract you need is not available, create a new authentication policy contract or edit an existing authentication policy contract in the **Policies** > **Authentication Policy Contracts** screen.
      - If the path is extended from a selector and it is the end of the path without any prior authentication source, select **--Continue--**, which leaves this path as an open-ended path.

        A 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.
5. Optional: Repeat *step 4* to create additional authentication policies.

Important: Order matters because the policy engine starts from the first policy and works its way down.

6. Optional: 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.

7. Click **Save**.

   If you are not ready to deploy all the policies yet, clear the **Enable IdP Authentication Policies** check box, the **Enable SP Authentication Policies**, or both, before saving your policies.

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 (also terminating the authentication policy, making it a closed-ended path)

  A 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.

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. Locate the authentication source that you want to define additional successful results for further processing.
2. Click **Rules** underneath **Success** (result).

   The PingFederate administrative console should display a **Rules** dialog.
3. Select an attribute from the previous authentication source in 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 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. Optional: Click **Delete** when a specific rule is no longer required.
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.

    The PingFederate administrative console brings you back to the **Manage Authentication Policies** screen. The authentication policy is updated with new successful results based on your 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**
    - **Success - Contractors** (a new result based on the first rule)
    - **Success - Senior executives** (a new result based on the second rule)
    - **Success** (available only when the **Default to Success** check box is selected)

    Select an action for each result.

Specify an incoming user ID

Some authentication sources, such as the PingID™ Adapter, require a user ID to be passed in from an earlier-authentication step to perform multifactor authentication. Additionally, all IdP adapters developed using the `IdpAuthenticationAdapterV2` interface from the PingFederate SDK (including the bundled HTML Form Adapter) and SAML 2.0 IdP connections supporting the SP-initiated Browser SSO profile may support an incoming user ID.

For these use cases, use the **Options** > **Incoming User ID** dialog to specify the incoming user ID in your authentication policies.

In a nutshell, for the authentication source that needs an incoming user ID, you select the source and the attribute in the policy; the source can be any IdP adapter instance, an IdP connection that is capable of handling SP-initiated Browser SSO requests, or the original SAML 2.0 AuthnRequest. At runtime, PingFederate maps the attribute value as the user ID.

1. Locate the authentication source that you need to provide an incoming user ID, and then click **Options** (underneath the entry).

   The **Incoming User ID** dialog should open.
2. Select the source of the attribute from the list.

   You can choose from any IdP adapter instances and IdP connections that were placed ahead of the current authentication source in the same path or `Context`, which represents the original SAML 2.0 AuthnRequest at runtime.
3. Select an attribute from the list.

If you have selected `Context` from the **Source** list, select the `Requested User` attribute, which represents the optional `subject` identifier in the SAML 2.0 AuthnRequest at runtime.

If a request does not originate with an SAML 2.0 AuthnRequest (for example, an IdP-initiated SSO request or a WS-Federation SSO request), or if a SAML 2.0 AuthnRequest does not include the optional `subject` identifier, the policy engine does not pre-populate the user ID.

4. Click **Done** to close the **Incoming User ID** dialog.

   The PingFederate administrative console brings you back to the **Manage Authentication Policies** screen.

Apply policy contracts 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 policy contract for different closed-ended paths (in one or more authentication policies) and fulfill them differently to suit the requirements. For IdP Browser SSO use cases, the same policy contract can also be mapped to multiple SP connections, thus enforcing the same set of authentication policies to those connections.

To apply an authentication policy contract to a policy, select an authentication policy contract as the last action of one or more paths and configure fulfillment for each contract.

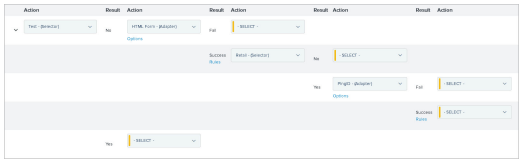
1. Locate all closed-ended paths in the policy.

   A 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.

   A 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.

   You cannot map an authentication policy contract to an open-ended path because the users have not yet been challenged by any authentication source.

   Consider the following sample policy:

   This policy has five paths:

   - **Test (Selector)** > **No** > **HTML Form (Adapter)** > **Fail**
   - **Test (Selector)** > **No** > **HTML Form (Adapter)** > **Success** > **Retail (Selector)** > **No**
   - **Test (Selector)** > **No** > **HTML Form (Adapter)** > **Success** > **Retail (Selector)** > **Yes** > **PingID (Adapter)** > **Fail**
   - **Test (Selector)** > **No** > **HTML Form (Adapter)** > **Success** > **Retail (Selector)** > **Yes** > **PingID (Adapter)** > **Success**
   - **Test (Selector)** > **Yes**

   The first four paths are closed-ended while the last path is open-ended.

2. For each applicable closed-ended path, select the applicable authentication policy contract as the last action from the **Action** list.

   It makes sense to select an authentication policy contract for the **PingID (Adapter)** > **Success** result, because the users have successfully met all your authentication requirements. Depending on your use case, you may also select an authentication policy contract for the **PingID (Adapter)** > **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 workflow in the applicable connections later.

   If the authentication policy contract you need is not available, create a new authentication policy contract or edit an existing authentication policy contract in the **Policies** > **Authentication Policy Contracts** screen.

3. For each selected authentication policy contract, click **Contract Mapping** and configure the authentication policy contract using the **Manage Authentication Policies** > **Authentication Policy Contract Mapping** wizard.
   a) Optional: Click **Add Attribute Source** in the **Attribute Sources & User Lookup** screen to configure data store queries.
   b) In 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 sample policy, 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.
   c) Optional: In the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract. (For more information, see *Define issuance criteria for authentication policy contract mapping*.)
   d) In the **Summary** screen, review your configuration, modify as needed, and then click **Done**.

      The PingFederate administrative console brings you back to the **Manage Authentication Policies** screen.

Define issuance criteria for authentication policy contract mapping

Use the **Issuance Criteria** screen to define criteria that PingFederate can evaluate to determine whether to approve or reject requests for protected resource. (For more information about this optional workflow, see *About token authorization*.)

The configuration begins by selecting a source containing the attribute to be verified. Some sources, such as **Context** and **Mapped Attributes**, are common to almost all use cases. Other sources, such as **Assertion**, have dependency on the type of configuration, for which the administrative console automatically hides the irrelevant sources automatically. After selecting a source, choose an attribute to be verified. Depending on the selected source, attributes vary. For example, if **Account Linking** is the selected source, **Local User ID** is the available attribute. Finally, select the comparison method, specify the desired value, and enter an optional error message for scenarios where the condition fails.

When multiple criteria are configured through the user interface, all criteria must be met at runtime (evaluated as *true*) for the request to succeed; otherwise PingFederate rejects the request and returns an error message. If only a subset or a layer of the criteria is required, use one or more OGNL expressions to define them.

1. Select a source containing the attribute to be verified.

   For more information about the **Source** list, refer to the following table:

Source	Description
Adapter	Select to access attributes from any IdP adapter instance leading up to this authentication policy contract.
IdP Connection	Select to access attributes from any IdP connection leading up to this authentication policy contract.
JDBC, LDAP, or Custom (if configured)	Select to access attributes returned from a data source.
Mapped Attributes	Select to access the mapped attributes.

2. Select an attribute from the **Attribute Name** list.
3. Select a comparison method from the **Condition** list.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value in the text field.

   Note: If your use case requires more complex evaluations, such as partial matching, use one or more OGNL expressions to define them.

5. Optional: Enter a custom error message in the **Error Result** field that PingFederate returns when the criterion fails.

   If you leave this field blank, PingFederate returns `ACCESS_DENIED` at runtime when the criterion fails.
6. Click **Add**.

   Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the criterion.
7. Optional: Repeat these steps to specify additional criteria.

Define issuance criteria using OGNL expressions

1. Click **Show Advanced Criteria**.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Attribute mapping expressions*).
2. Specify a criterion.
   a) Enter an OGNL expression into the text field.
   b) Enter a custom error message in the **Error Result** field.

      Note: If the OGNL expression resolves to a string value (rather than *true* or *false*), the returned value overrides the custom error message.
   c) Click **Add**.
   d) Click the **Test** link to input values and test the resulting output for the expression.

   Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the OGNL expression.
3. Optional: Repeat these steps to specify another OGNL expression.

   If multiple OGNL expressions are specified, they all must pass in order for the request to succeed. Alternatively, you can combine multiple criteria in one OGNL expression.

Map a policy contract to multiple connections

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.
2. In 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.

Configure an SP authentication policy for users from one IdP

For SP-initiated Browser SSO requests received at the `/sp/startSSO.ping` application endpoint, PingFederate provides the optional capability to enforce additional authentication requirements through the use of SP authentication policies. (For general concepts and runtime behavior of authentication policies, see *Policies*.)

When you enable SP authentication policies to enforce additional authentication requirements, you are configuring policies for the policy engine in the PingFederate SP server to find an applicable SP adapter instance to access your target applications. Therefore, you must configure the target applications to provide the `SpSessionAuthnAdapterId` parameter, the `TargetResource` parameter, or both, when submitting their SP-initiated SSO requests to the `/sp/startSSO.ping` application endpoint. If you prefer to provide the `TargetResource` parameter *without* the `SpSessionAuthnAdapterId` parameter, you must configure one or more entries in the **SP Configuration** > **Target URL Mapping** screen to map the `TargetResource` values to the applicable SP adapter instances.

Note: When SP authentication policies are turned on, they are applicable only to SP-initiated Browser SSO requests received at the `/sp/startSSO.ping` application endpoint and not unsolicited SSO requests (namely IdP-initiated Browser SSO requests via SAML 1.x or 2.0).

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 **SP Configuration** > **Adapters** wizard and completed the last-mile integration with the target application. The SP adapter name and ID are **OpenToken SpSample** and `opentokenSpSample`, respectively. In the **SP Configuration** > **Server Settings** > **Federation Info** screen, the base URL for your PingFederate SP server is defined as `https://sso.xray.local:9071`. 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-initated SSO URL (*step 6*).

To fulfill the requirements:

1. In the **SP Configuration** 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) In the **IdP Connection** > **Browser SSO** > **SAML Profiles** screen, make sure that the **SP-Initiated SSO** check box is selected.
   c) In the **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Identity Mapping** screen, select **No Mapping**.

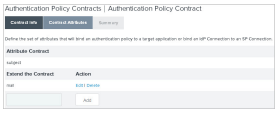
      Tip: If the partner and your organization agree to support account linking, select **Account Linking**. If the partner and your organization agree to support the IdP-initiated profile, select **Account Mapping** or **Account Linking**.

      Both use cases require the applicable SP adapter instance (**OpenToken SpSample**) 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.
   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. Go to **SP Configuration** > **Policies** screen. In the **Authentication 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) Follow the wizard to create an authentication policy contract.
   b) In the **Contract Attributes** screen, extend the contract two attributes, `subject` and `mail`; for example:

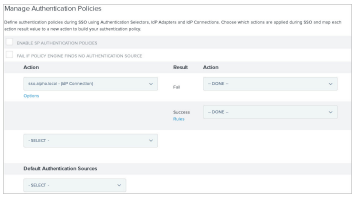


c) Complete the rest of the connection configuration.

4. Go to the **SP Configuration** > **Policies** screen to define a policy to enforce third-party authentication.

a) Select **sso.alpha.local (IdP Connection)** from the **Action** list.

In general, an IdP connection has two results, **Fail** and **Success**; for example:

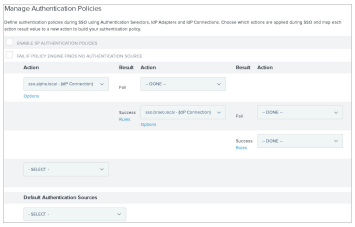


Each result forms its own path that requires further configuration

b) For the **sso.alpha.local (IdP connection)** > **Fail** path, leave its action as **--Done--**.

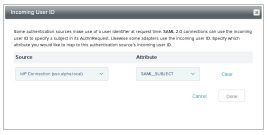
The policy engine stops. PingFederate rejects the SSO request.

c) For the **sso.alpha.local (IdP connection)** > **Success** path, select **sso.bravo.local (IdP Connection)** from its **Action** list (which also forms two paths); for example:



d) Click **Options** (underneath **sso.bravo.local (IdP Connection)**) to relay the user ID (`SAML_SUBJECT`) from `sso.alpha.local` to `sso.bravo.local`.

In the **Incoming User ID** dialog, select **IdP Connection (sso.alpha.local)** from the **Source** list and **SAML_SUBJECT** from the **Attribute** list; for example:

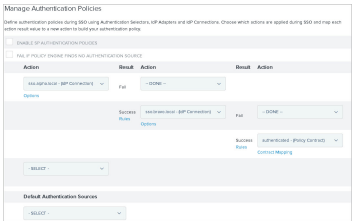


Click **Done**.

e) For the **sso.bravo.local (IdP connection)** > **Fail** path, leave its action as **--Done--**.

The policy engine stops. PingFederate rejects the SSO request.

f) For the **sso.bravo.local (IdP connection)** > **Success** path, select the authentication policy contract created in *step 3*; for example:



g) Click **Contract Mapping** (underneath the authentication policy contract) and follow the **Manage Authentication Policies** > **Authentication Policy Contract Mapping** wizard to configure the fulfillment of the authentication policy contract.

Optional: In the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries. (For more information, see *Fulfillment by data store queries*.)

In the **Contract Fulfillment** screen, select **IdP Connection (sso.alpha.local)** from the **Source** list and the appropriate attributes in the **Value** list to fulfill the authentication policy contract attributes; for example:

Tip: Essentially, you are mapping attributes to an authentication policy contract from the authentication policy.

Optional: In the **Issuance Criteria** screen, configure conditions to be validated before issuing an authentication policy contract. (For more information, see *Define issuance criteria for authentication policy contract mapping*.)

Click **Done** and then **Save**.

5. Go to **SP Configuration** > **Adapter Mappings** screen. In the **Authentication Policy 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 from the **Target Instance** list.
   c) Click **Add Mapping**.
   d) Follow the **Authentication Policy Adapter Mappings** > **Mapping Configuration** wizard to create the mapping.

   Optional: In the **Attribute Sources & User Lookup** screen, click **Add Attribute Source** to configure data store queries. (For more information, see *Fulfillment by data store queries*.)

   Optional: In the **Target App Info** screen, enter application information.

   In the **Adapter Contract Fulfillment** screen, select **Authentication Policy Contract** from the **Source** list and the appropriate contract attributes in the **Value** list to fulfill the SP adapter contract; for example:

   Tip: Basically, you are mapping attribute values from the authentication policy contract to the target application through the applicable SP adapter instance.

   Optional: In the **Default Target URL** screen, specify a default target URL for this mapping configuration.

   Optional: In the **Issuance Criteria** screen, configure conditions to be validated before issuing an SP adapter contract. (For more information, see *Define issuance criteria for adapter mapping*.)

   Click **Done**.

6. Configure an SP-initated SSO URL in your target application by combining the base URL of your PingFederate SP server (`https://sso.xray.local:9071`), 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:9071/sp/startSSO.ping?
   SpSessionAuthnAdapterId=opentokenSpSample
   ```

   If you have not defined a default URL for the adapter mapping (configured in *step 4*), the IdP connection, or the PingFederate SP server, you must also configure your target application to include the `TargetResource` parameter in its SP-initated 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 enable SP authentication policies, go to the **SP Configuration** > **Policies** screen, select the **Enable SP Authentication Policies** check box, 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.

Suppose you configured the following use cases in PingFederate 8.0:

- Two SP adapter instances in the **SP Configuration** > **Adapters** screen:

Tip: In this example, because the applications use the same SP adapter type, the same configuration, and the same set of user attributes (`subject` and `spEmail`), **Sample Delta** is a child adapter of **Sample** without any overridden settings.

- Three entries in the **SP Configuration** > **Target URL Mapping** screen:

- Three IdP connections to your partners as follows:

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`
		
		`email`
			Sample
			
			(`sample`)

Charlie
 	
(`sso.charlie.local`)
	Account Mapping	`SAML_SUBJECT`
		
		`email`
			Sample
			
			(`sample`)

Delta
 	
(`sso.delta.local`)
	Account Mapping	`SAML_SUBJECT`
		
		`email`
			Sample Delta
			
			(`sampleDelta`)

Note that all partners support SAML 2.0 and only the SP-initiated SSO profile.

- SSO URLs for users from Alpha, Charlie, and Delta, as follows:

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%2F%3Fapp%3DAlpha%26tab%3Dhome`

Partner	SSO URL
Charlie	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.charlie.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%2F%3Fapp%3DCharlie%26tab%3Dhome`
Delta	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.delta.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%2F%3Fapp%3DDelta%26tab%3Dhome`

After upgrading to PingFederate 8.1, 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, which 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` (without other attributes)	N/A
Echo (`sso.echo.local`)	No Mapping	`SAML_SUBJECT` `email`	N/A
Foxtrot (`sso.foxtrot.local`)	Account Mapping	`SAML_SUBJECT` `email`	Sample (`sample`)
Golf (`sso.golf.local`)	No Mapping	`SAML_SUBJECT` `email`	N/A

  Note that 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 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. Go to the **SP Configuration** > **Adapters** screen to create two new SP adapter instance: **Sample Echo** and **Sample Golf**; for examples:

   Tip: In this example, because the applications use the same SP adapter type, the same configuration, and the same set of user attributes, **Sample Echo** and **Sample Golf** are child adapters of **Sample** without any overridden settings.

2. Create an IdP connection to Bravo.
   a) In the **SP Configuration** menu, click **Create New** (under **IdP Connections**) and follow the **IdP Connection** wizard to create a SAML 2.0 IdP connection.
   b) In 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**.
   d) Complete the rest of the connection configuration.
3. Create IdP connections to Echo and Golf.
   a) In the **SP Configuration** 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) In the **IdP Connection** > **Browser SSO** > **SAML Profiles** screen, select the **SP-Initiated SSO** check box.

      Tip: If the partner and your organization agree to support other profiles, select them accordingly.

   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**. 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 Echo** or **Sample Golf**) 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.
   d) In the **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Attribute Contract** screen, extend the contract with the `email` attribute.
   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) In the **SP Configuration** menu, click **Create New** (under **IdP Connections**) and follow the **IdP Connection** wizard to create a SAML 2.0 IdP connection.
   b) In the **IdP Connection** > **Browser SSO** > **SAML Profiles** screen, select the **SP-Initiated SSO** check box.

      Tip: If the partner and your organization agree to support other profiles, select them accordingly.

   c) In the **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Identity Mapping** screen, select **Account Mapping**.

d) In the **IdP Connection** > **Browser SSO** > **User-Session Creation** > **Attribute Contract** screen, extend the contract with the `email` attribute.

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** wizard to map the attributes from the assertion to the SP adapter instance **Sample**.

f) Complete the rest of the connection configuration.

5. Create three authentication policy contracts.

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 7* in this example).
- You map attributes from authentication policy contracts to target applications through adapter mappings (*step 9* in this example).

a) Go to the **SP Configuration** > **Policy Contracts** screen, click **Create New Contract** and follow the **Authentication Policy Contract** wizard to create an authentication contract for users from Alpha and Charlie (Echo, and then Golf).

b) In the **Contract Attributes** screen, extend the authentication policy contract with an attribute for user's email address; for example, `apcEmail`.

c) Complete the rest of the connection configuration.

d) Repeat these steps to create an authentication policy contract for users from Echo, and then for users from Golf; for examples:

6. **Note:** If multiple target applications share the same SP adapter instance, you must use a selector to evaluate the SP-initated 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. Based on the results, the policy diverts accordingly.

Create an instance of the HTTP Request Parameter Authentication Selector.

a) Go to the **SP Configuration** > **Selectors** screen and click **Create New Instance**.

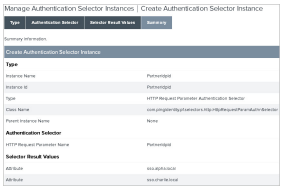
b) In the **Type** screen, select **HTTP Request Parameter Authentication Selector** from the **Type** list and provide a name and ID for the selector instance.

c) In the **Authentication Selector** screen, enter `PartnerIdpId` in the **HTTP Request Parameter Name** field.

d) In 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:



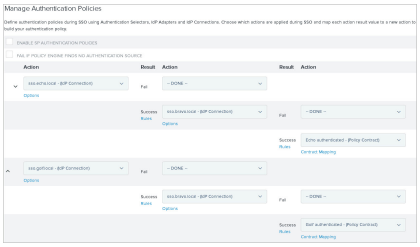
7. Go to the **SP Configuration** > **Policies** screen to define a new authentication policy to enforce third-party authentication for users from Echo (and then for users from Golf).

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 222.

a) Select **sso.echo.local (IdP Connection)** from the **Action** list.
b) For the **sso.echo.local (IdP connection)** > **Fail** path, leave its action as **--Done--**.
c) For the **sso.echo.local (IdP connection)** > **Success** path, select **sso.bravo.local (IdP Connection)** from its **Action** list.
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 (IdP connection)** > **Fail** path, leave its action as **--Done--**.
f) For the **sso.bravo.local (IdP connection)** > **Success** path, select the authentication policy contract **Echo authenticated** from its **Action** list, and then click **Contract Mapping** (underneath the authentication policy contract) to configure the fulfillment of the authentication 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 examples:

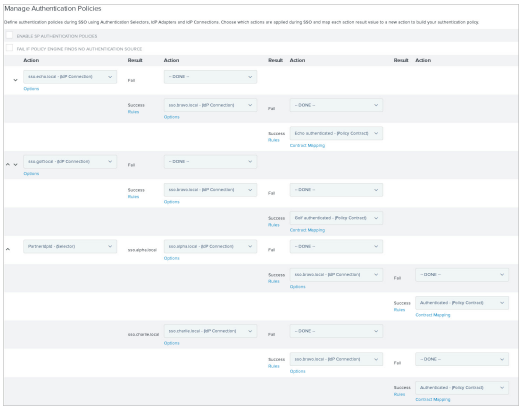


8. In the **Manage Authentication Policies** screen, define a new authentication policy to enforce third-party authentication for users from Alpha and Charlie.

Note: If multiple target applications share the same SP adapter instance, you must use a selector to evaluate the SP-initated 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) Select the instance of the HTTP Request Parameter Authentication Selector from the **Action** list.
b) For the path with result of **sso.alpha.local**, repeat *step 7* to configure the authentication requirement for the `sso.alpha.local` IdP connection.
c) For the path with result of **sso.charlie.local**, repeat *step 7* to configure the authentication requirement for the `sso.charlie.local` IdP connection.

At this point, the SP authentication policies are configured as follows:

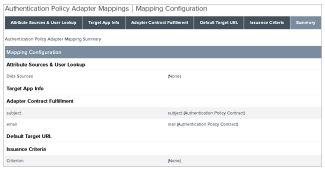


d) Click **Save**.

9. Create adapter mappings.
a) Go to the **SP Configuration** > **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 (**Sample**, **Sample Echo**, and **Sample Golf**); for examples:

For each adapter mapping:



Tip: Basically, 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 **SP Configuration** > **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**

At this point, the target URL mappings are configured as follows:

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%2F%3Fapp%3DAlpha%26tab%3Dhome` 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%2F%3Fapp%3DCharlie%26tab%3Dhome` 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.

Partner	SSO URL
Delta	`https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.delta.local&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%2F%3Fapp%3DDelta%26tab%3Dhome` 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%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%2F%3Fapp%3DEcho%26tab%3Dhome` 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?SpSessionAuthnAdapterId=sample&TargetResource=https%3A%2F%2Fsso.xray.local%3A9031%2FSpSample%2FMainPage%2F%3Fapp%3DFoxtrot%26tab%3Dhome` 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%2F%3Fapp%3DGolf%26tab%3Dhome` 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 enable SP authentication policies, go to the **SP Configuration** > **Policies** screen, select the **Enable SP Authentication Policies** check box, 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.

For example, you configured the following use cases in PingFederate 8.0.

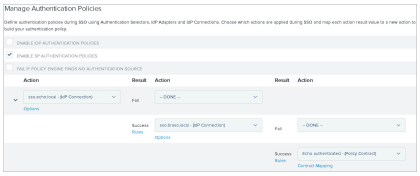
For users from Echo (an IdP):

- An SP adapter instance:
  - Name: Sample Echo
  - ID: `sampleEcho`
- A WS-Federation IdP connection:
  - Partner: Echo
  - Federation ID: `sso.echo.local`
  - Identity mapping method: Account mapping
  - Default target URL: `https://sso.xray.local:9031/SpSample/MainPage/?app=Echo&tab=home`
  - SSO URL: `https://sso.xray.local:9031/sp/startSSO.ping?PartnerIdpId=sso.echo.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 Echo
  - Default target URL: `https://sso.xray.local:9031/SpSample/MainPage/?app=Internal&tab=home`
  - SSO URL: `https://sso.xray.local:9031/pf/adapter2adapter.ping?SpSessionAuthn=sampleEcho`

After upgrading to PingFederate 8.1, if you want to enforce multifactor authentication for users from Echo through Bravo, you can create an IdP connection to Bravo and the following authentication policy:



Because the authentication policy ends with a policy contract **Echo authenticated**, you must create an adapter mapping (from the policy contract) to **Sample Echo**, the SP adapter instance integrated with the target application. You also need to update the SSO URL for users from Echo to:

`https://sso.xray.local:9031/sp/startSSO.ping?SpSessionAuthnAdapterId=sampleEcho`

When you select the **Enabled SP Authentication Policies** check box to turn on the policy, the `/pf/adapter2adapter.ping` endpoint is disabled to prevent malicious Echo'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. Go to the **SP Configuration** > **Adapters** screen to create a new SP adapter instance; for example, **Sample Internal** (with an SP adapter ID of `sampleInternal`).

   Tip: In this example, because the applications use the same SP adapter type, the same configuration, and the same set of user attributes (`subject` and `spEmail`), **Sample Internal** is a child adapter of **Sample Echo** without any overridden settings.

2. Go to the **SP Configuration** > **Policy Contracts** screen to create a new authentication policy contract; for example, **Internal authenticated**.

   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).

3. Go to the **SP Configuration** > **Selectors** screen to create an instance of the CIDR Authentication Selector with one or more network ranges that correspond to your 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 **SP Configuration** > **Policies** screen to configure your policies as follows.

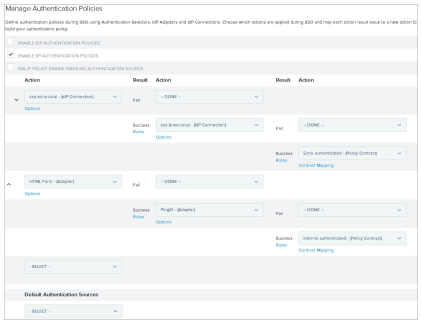


Figure 23: With PingID™ Adapter

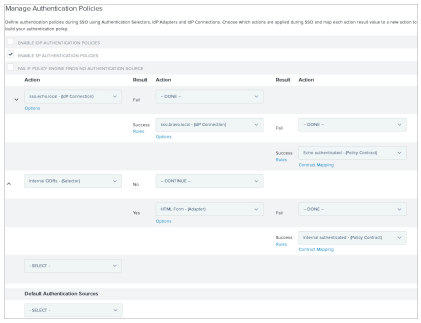


Figure 24: With CIDR Authentication Selector

5. Go to the **SP Configuration** > **Adapter Mappings** screen to create a mapping from the new authentication policy contract (Internal authenticated) to the new SP adapter instance (`sampleInternal`).

   Enter `https://sso.xray.local:9031/SpSample/MainPage/?app=Internal&tab=home` in the **Default Target URL** screen.
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. In other words, 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 or IdP connections.

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* on page 83).

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 path, and to apply the authentication policy contract to the target application.

Configure authentication policy adapter mappings

1. Go to **SP Configuration** > **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: In 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 494.
   b) Optional: In the **Target App Info** screen, enter application information.
   c) In 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.
   d) Optional: In the **Default Target URL** screen, specify a default target URL for this mapping configuration.
   e) Optional: In the **Issuance Criteria** screen, configure conditions to be validated before issuing an SP adapter contract (see *Define issuance criteria for adapter mapping*).
   f) In the **Summary** screen, review the configuration and modify as needed. When complete, click **Done**.
6. In the **Authentication Policy Adapter Mappings** screen, click **Save**.

Define issuance criteria for adapter mapping

Use the **Issuance Criteria** screen to define criteria that PingFederate can evaluate to determine whether to approve or reject requests for protected resource. (For more information about this optional workflow, see *About token authorization*.)

The configuration begins by selecting a source containing the attribute to be verified. Some sources, such as **Context** and **Mapped Attributes**, are common to almost all use cases. Other sources, such as **Assertion**, have dependency on the type of configuration, for which the administrative console automatically hides the irrelevant sources automatically. After selecting a source, choose an attribute to be verified. Depending on the selected source,

attributes vary. For example, if **Account Linking** is the selected source, **Local User ID** is the available attribute. Finally, select the comparison method, specify the desired value, and enter an optional error message for scenarios where the condition fails.

When multiple criteria are configured through the user interface, all criteria must be met at runtime (evaluated as *true*) for the request to succeed; otherwise PingFederate rejects the request and returns an error message. If only a subset or a layer of the criteria is required, use one or more OGNL expressions to define them.

1. Select a source containing the attribute to be verified.

   For more information about the **Source** list, refer to the following table:

Source	Description
Authentication Policy	Select to access attributes from the authentication policy contract.
Context	Select to use values 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)	Select to access attributes returned from a data source.
Mapped Attributes	Select to access the mapped attributes.

2. Select an attribute from the **Attribute Name** list.
3. Select a comparison method from the **Condition** list.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value in the text field.

   Note: If your use case requires more complex evaluations, such as partial matching, use one or more OGNL expressions to define them.
5. Optional: Enter a custom error message in the **Error Result** field that PingFederate returns when the criterion fails.

   If you leave this field blank, PingFederate returns `ACCESS_DENIED` at runtime when the criterion fails.
6. Click **Add**.

   Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the criterion.
7. Optional: Repeat these steps to specify additional criteria.

Define issuance criteria using OGNL expressions

1. Click **Show Advanced Criteria**.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Attribute mapping expressions*).
2. Specify a criterion.
   a) Enter an OGNL expression into the text field.
   b) Enter a custom error message in the **Error Result** field.

      Note: If the OGNL expression resolves to a string value (rather than *true* or *false*), the returned value overrides the custom error message.
   c) Click **Add**.
   d) Click the **Test** link to input values and test the resulting output for the expression.

   Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the OGNL expression.

3. Optional: Repeat these steps to specify another OGNL expression.

   If multiple OGNL expressions are specified, they all must pass in order for the request to succeed. Alternatively, you can combine multiple criteria in one OGNL expression.

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. You must configure Server Settings from the Main Menu to establish your site as an IdP before configuring connections to SPs (see *Choose roles and protocols* on page 136).

Note that only one connection is needed per partner, even if you are targeting more than one web application at the destination SP site. You can configure more than one connection, however, if your partner supports multiple protocols, or supports multiple federation IDs for the same protocol (see *Federation Server Identification*).

Note: This chapter 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 378.

Under some conditions, you can enable SSO for an unlimited number of partners at once by configuring a single, common connection (see *Using Auto-Connect* on page 75).

You can also 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* on page 83 and *Deploying PingFederate as a federation hub* on page 86 for more information).

This chapter covers the following topics:

- *IdP application integration settings* on page 238
- *View IdP protocol endpoints* on page 244
- *Manage SP connections* on page 245
- *Define SP affiliations* on page 305
- *Configure SP Auto-Connect* on page 306

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 64).

Under Application Integration Settings on the IdP Configuration menu, you configure the IdP Adapters 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 Settings also include links for configuring plug-in **Token Processors** and, optionally, **STS Request Parameters**. Locate configuration information under *IdP configuration for STS* on page 381.

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 following adapters:

- Kerberos Adapter
- HTML Form Adapter
- HTTP Basic Adapter
- OpenToken Adapter

- Composite Adapter

You can also deploy additional integration kits from *Ping Identity web site* (www.pingidentity.com/en/products/downloads/pingfederate.html).

You manage IdP adapter instances in the **IdP Configuration** > **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**.

Create an IdP adapter instance

The first step in creating an adapter instance is choosing an adapter type. Some adapters are bundled with PingFederate. Other adapters and integration kits are available from the *Ping Identity web site* (www.pingidentity.com/en/products/downloads/pingfederate.html).

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** screens 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.

- For adapters packaged with PingFederate, refer to the following documentation for configuration instructions:
  - *Kerberos Adapter*
  - *HTML Form Adapter*
  - *HTTP Basic Adapter*
  - *OpenToken Adapter*
  - *Composite Adapter*
- If you have deployed an adapter from an integration kit, locate the user guide from *Knowledge Center* at *pingidentity.com* and configure the adapter instance accordingly.
- Click **Show Advanced Fields** to enter an authentication context URI for an adapter that supports this feature.

  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 selected adapter supports authentication context. (For a discussion of authentication context, find that term under *Terminology*.)

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.

  Tip: For information about actions available using the OpenToken Adapter, see *Configure the OpenToken IdP Adapter*.

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 (see *Configure the Composite Adapter*).

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 using the **Adapter Contract Mapping** workflow later.

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 workflow.

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 workflow. 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**.

For more information, see *Fulfillment by data store queries* .

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.
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. For more information, see *Attribute mapping expressions*.
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

Use the **Issuance Criteria** screen to define criteria that PingFederate can evaluate to determine whether to approve or reject requests for protected resource. (For more information about this optional workflow, see *About token authorization*.)

The configuration begins by selecting a source containing the attribute to be verified. Some sources, such as **Context** and **Mapped Attributes**, are common to almost all use cases. Other sources, such as **Assertion**, have dependency on the type of configuration, for which the administrative console automatically hides the irrelevant sources automatically. After selecting a source, choose an attribute to be verified. Depending on the selected source, attributes vary. For example, if **Account Linking** is the selected source, **Local User ID** is the available attribute. Finally, select the comparison method, specify the desired value, and enter an optional error message for scenarios where the condition fails.

When multiple criteria are configured through the user interface, all criteria must be met at runtime (evaluated as *true*) for the request to succeed; otherwise PingFederate rejects the request and returns an error message. If only a subset or a layer of the criteria is required, use one or more OGNL expressions to define them.

1. Select a source containing the attribute to be verified.

   For more information about the **Source** list, refer to the following table:

Source	Description
Adapter	Select to access attributes from the IdP Adapter.
Context	Select to use values 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)	Select to access attributes returned from a data source.
Mapped Attributes	Select to access the mapped attributes.

2. Select an attribute from the **Attribute Name** list.
3. Select a comparison method from the **Condition** list.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value in the text field.

   Note: If your use case requires more complex evaluations, such as partial matching, use one or more OGNL expressions to define them.
5. Optional: Enter a custom error message in the **Error Result** field that PingFederate returns when the criterion fails.

   If you leave this field blank, PingFederate returns `ACCESS_DENIED` at runtime when the criterion fails.
6. Click **Add**.

   Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the criterion.
7. Optional: Repeat these steps to specify additional criteria.

Define issuance criteria using OGNL expressions

1. Click **Show Advanced Criteria**.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Attribute mapping expressions*).
2. Specify a criterion.

a) Enter an OGNL expression into the text field.
b) Enter a custom error message in the **Error Result** field.

Note: If the OGNL expression resolves to a string value (rather than *true* or *false*), the returned value overrides the custom error message.

c) Click **Add**.
d) Click the **Test** link to input values and test the resulting output for the expression.

Use the **Edit**, **Update**, **Cancel**, **Delete**, and **Undelete** links to manage the OGNL expression.

3. Optional: Repeat these steps to specify another OGNL expression.

If multiple OGNL expressions are specified, they all must pass in order for the request to succeed. Alternatively, you can combine multiple criteria in one OGNL expression.

Review an adapter contract

- On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Adapter Contract Mapping** workflow.

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 page, 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 *Localization* on page 124. 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 442), but 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 120).

View IdP application endpoints

Click **Application Endpoints** on the IdP Configuration menu to see a list of endpoints and descriptions applicable to your federation role (IdP or SP). These endpoints are built into PingFederate and cannot be changed.

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate (see *SSO integration kits and adapters* on page 64).

Note: For specific parameters required or allowed for Application Endpoints, see *IdP endpoints* on page 442.

This screen also shows a Maintenance Endpoint that you can use to verify that the PingFederate server is running (see *System-services endpoints* on page 452).

View IdP protocol endpoints

Click Protocol Endpoints under Federation Settings in the IdP Configuration section of the Main Menu to see a list of SAML, WS-Federation, and/or WS-Trust STS endpoints—a pop-up window displays only those endpoints related to the federation protocols enabled in Server Settings (see *Choose roles and protocols* on page 136). These endpoints are built into PingFederate and cannot be changed.

PingFederate provides a favorite icon for all Protocol Endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 128.

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 (see *Export metadata to an XML file* on page 101).

The table below describes each endpoint:

Table 12: PingFederate IdP Endpoints

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**" footnote in this table.)
Attribute Query Service (SAML 2.0)	`/idp/attrsvc.ssaml2` The SAML implementation that receives and processes attribute requests. (See "**Important**" footnote in this table.)
Metadata Service	`/` The default endpoint (empty path) from which partners can retrieve Auto-Connect metadata (see *Using Auto-Connect* on page 75).
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**" footnote in 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

Service	URL and Description
	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 421). **Note:** If multiple token-processor instances of the same type are configured for the same connection or token-to-token mapping, a query parameter, `TokenProcessorId`, must be added to either of these endpoints—see *Manage token processors* on page 381. (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 *PingFederate properties* on page 112).

Note: For any connection using multiple virtual server IDs (see *Multiple virtual server IDs* on page 81), each virtual server ID has its own set of protocol endpoints. Use Metadata Export on the Server Configuration menu to export a connection metadata for your partner. For more information, see *Export metadata to an XML file* on page 101.

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 IdP or STS client application at your site.

Note: If you are configuring a new connection only for WS-Trust STS, follow the sections in this part of the manual up to and including *Identify the SP* on page 253. Then turn to *WS-Trust STS configuration* on page 378.

These settings include:

- User attributes you expect to send in an SSO assertion (including STS SAML tokens).
- User attributes that may be sent using the Attribute Query profile (if that profile is used).
- The protocol and, for SAML, the profile you will use, including detailed security specifications (the use of digital signatures, signature verification, XML encryption, and SSL). For more information, see *Supported standards* on page 28 in the *Get started with PingFederate* guide.

To continue with the configuration, you and your connection partner must have decided this information in advance (see *Federation planning checklist* on page 79). Your federation partner must supply some connection settings and other information (see *Configuration data exchange* on page 82).

Tip: If you are configuring connections to more than one partner under SAML 2.0 specifications, or if you intend to add partners in the future, consider using Auto-Connect (see *Configure SP Auto-Connect* on page 306).

If your agreement includes sending assertions containing attribute values from a local data store, then you need to define the data store during this configuration if you have not done so already (see *Manage data stores* on page 143).

Access SP connections

You can create, modify or import connections directly under SP Connections. Note that the IdP Configuration menu displays the four most-recently modified connections. To view a list of all SP connections, click the **Manage All** button.

Via the Main Menu

From the Main Menu under IdP Configuration, you can configure a new connection, modify an existing connection, import a connection, or view connections.

Tip: To copy, delete or export connections, as well as to recover connection drafts, click **Manage All** (see *Via the Manage Connections screen* on page 246).

Note that long connection names are truncated for this display and the list is limited to four connections, chronologically ordered according to most recently edited. The full connection names and a complete list are displayed on the Manage Connections screen (see *Via the Manage Connections screen* on page 246).

To begin configuring a new connection:

1. Click **IdP Configuration** on the Main Menu.
2. Click **Create New** under SP Connections.

Tip: The fastest way to create a new connection is to click **Manage All** and copy the connection with similar settings (see *Via the Manage Connections screen* on page 246).

To modify a connection

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Twenty-five most recently edited connections are displayed. To see all connections, including drafts, click **Manage All**.
3. On the Activation & Summary screen, click the heading for the information you want to change.
4. Make your change and click **Save**.

Note: If **Save** is not available, it means your modification requires other changes or you are editing a screen that is part of a series of subtasks. Click **Next** and continue making indicated changes. The **Done** button indicates that further changes in the task are optional. When you have no further changes, click **Done** and then click **Save** on the task summary screen.

To import a connection:

1. Click **Import**.
2. In 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**.

Via the Manage Connections screen

From the Manage Connections screen you can:

- Create a new connection.
- Modify or copy an existing connection.
- Continue working on a connection draft.
- Delete a connection—if it is not active or referenced in other parts of the configuration (In Use).
- Export or import individual connection configurations.

  Note: The connection export function results in an XML file that you can modify and import into the same PingFederate server or another PingFederate server acting in the same federation role (IdP or SP) at your site. You can import connections in this screen. Alternatively, you can use the Connection Management Service to complete the task (see *Importing connections* on page 467). Finally, you also have the option to automate this process (see *Automating configuration migration* on page 114).
- Export metadata about a connection to expedite your partner's corresponding configuration (see *Export metadata to an XML file* on page 101).
- Update a SAML connection by using a metadata file or a metadata URL from your partner.

  Important: The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.

On this screen you can also globally override transaction logging levels set for individual connections or restore connection-based logging (see *Runtime transaction logging* on page 93).

Tip: A check-mark icon next to a Connection Name indicates that the connection has been checked for configuration errors. For more information about connection-validation features associated with this screen, see *Manage SP connection validation* on page 249.

To access the Manage Connections screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click **Manage All** under SP Connections.

To begin configuring a new SP connection:

- Click **Create Connection** on the Manage Connections screen.

To copy a connection:

1. Click **Copy** under Action for the connection you want to copy.
2. Update the Connection ID (of the partner) and the Connection Name in the General Info screen (see *Identify the SP* on page 253).
3. Make any further changes needed for the new connection.

Note: Outbound Provisioning configurations are not copied for a connection (see *Outbound provisioning for IdPs* on page 78).

To edit a connection or continue working on a draft:

- Click the Connection Name link.

  For a draft, you will return to where you left off.

To export a connection:

1. Click **Export Connection** under Action for the connection.
2. Save the XML file on your file system.

   You can change the name of the file, but keep the XML extension.

To import a connection:

1. Click **Import**.
2. In 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**.

To export connection metadata:

1. Click **Export Metadata** under Action for the connection.

   This action takes you to the Export Metadata screen flow, with the connection selection preset (see *Export metadata to an XML file* on page 101).
2. Complete the steps remaining in the Export Metadata screen flow (starting at *Step 4* under *Provide SAML metadata by file* on page 101).

To update a SAML connection with your partner's metadata:

1. Click **Update with Metadata** under Action for the applicable SAML connection.
2. Follow the workflow to complete the update process.

Important: The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.

To delete a connection:

1. Under Action, click **Delete** for the connection.

   (To undo the deletion, click **Undelete**.)

   Note: The **Delete** function is not available if the connection is Active or In Use.

2. To confirm the deletion, click **Save**.

To sort the list of connections:

- Click the arrow next to any column heading to sort the list based on that column.

Note: The Virtual ID displays the default virtual server ID of the connection, if any (see *Identify the SP* on page 253).

To filter the list by Protocol and/or Status:

- Select a filter criterion from either or both of the drop-down lists.

To override connection-based transaction logging:

1. Select **On** under Logging Mode Override.
2. Choose the logging mode you want to use for all connections.

To restore connection-based transaction logging:

- Select **Off** under Logging Mode Override.

Import a connection

Use the Import Connection screen to import a connection.

1. Click **Import**.
2. In 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 in the **Import Certificate** screen, and then click **Next**.

Metadata medium	Steps
	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** workflow, 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 in 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** workflow (see *Manage metadata URLs* on page 171). 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 **IdP Configuration** > **Manage All** > **Connections** screen.
2. Click **Update with Metadata** under Action for the applicable SAML connection.
3. Follow the workflow to complete the task.

Manage SP connection validation

By default PingFederate automatically validates all existing connections before displaying the Manage Connections screen. This validation ensures that any updates to supporting components—adapters or data-store configurations, for example—have not invalidated any connection settings.

If such errors are found, a warning icon appears next to the Connection Name.

To correct errors:

- Click the Connection Name to reach the top-level task in which reconfiguration is needed, and to see the error message. Then navigate into deeper tasks using **Configure . . .** buttons to find a link to the screen that needs updating.

(For more information about console navigation, see *About Tasks and Steps* in the "Console Navigation" chapter of the *Get started with PingFederate* guide).

Note that the connection validation time increases with the number of connections and when connections are configured to access data stores for attribute mapping. Consequently, there may be noticeable delays in displaying the Manage Connections screen. For this reason, PingFederate provides a way to turn off the automatic validation under Server Settings (see *Configure system options* on page 138).

When validation is turned off, administrators can check connections manually on the Manage Connections screen. A question-mark icon indicates the connection has not been validated. You may, however, still edit the connection by clicking its name.

In addition, Action links are disabled (except for **Delete**, if the connection is Inactive and/or In Use) until the connection is validated.

When automatic validation is disabled, use one of the following procedures to validate connections:

To validate a *single* connection:

- Click icon next to the Connection Name.

Note: Validating a single connection does not check connectivity for any configured data-store lookups (see *Select an attribute mapping method* on page 265). However, this check *is* performed when you access the connection for editing.

To validate *all* connections (including for data-store connectivity):

1. Click **Check All Connections for Errors**.
2. When prompted, click **Yes**.

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 (see *Outbound provisioning for IdPs* on page 78).

Tip: When you select a Connection Template, many connection settings are configured for you automatically. For information about using a template, refer to the *Quick Connection Guide* available with your PingFederate SaaS Connector.

- 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 preconfigures browser-based SSO), indicate on the Connection Type screen whether the connection to this partner is for Browser SSO, WS-Trust STS, and/or Outbound Provisioning (see *Connection types* on page 57).

Note: You can add STS, OAuth, and Outbound Provisioning support to any existing SSO connection, or vice versa, at any time.

If your federation deployment supports multiple protocols and you are not using a connection template, then for new SSO connections you can also select the applicable protocol on the Connection Type screen (see *Choose roles and protocols* on page 136).

Note: If your partner's deployment also supports multiple protocols and you intend to communicate using more than one, then you must set up a separate connection for each protocol.

- To configure a connection for secure browser-based SSO, select Browser SSO Profiles and the Protocol (if necessary). 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 a connection for WS-Trust STS, make that selection and then select a Default Token Type.

  The Default Token Type, either SAML 1.1 or 2.0, is used when a Web Service client does not specify in the token request what token type the STS should issue.

  Note: The Default Token Type *does not* need to match the Protocol indicated on the screen for SSO (when applicable).
- To configure a connection for Outbound Provisioning, make that selection and then select the Outbound Provisioning Type (if multiple outbound provisioning drivers are installed, such as SCIM, Google, or Salesforce).

  Note: The Outbound Provisioning option is active only after you enable the Outbound Provisioning protocol on the Roles & Protocols screen (see *Choose roles and protocols* on page 136).
- 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 SP connection options

On the Connection Options screen, you can enable the Browser SSO and Attribute Query options for the current connection. In addition, for browser-based SSO you can enable IdP Discovery for the current connection.

Note: This screen is presented only for browser-based SSO connections (see *Choose an SP connection type* on page 250).

- To create a connection for browser-based SSO, select Browser SSO.
- To enable IdP Discovery for this connection, select IdP Discovery.

  Note: The IdP Discovery check box is only enabled if IdP Discovery is configured and the Domain Cookie Setting "IdP using a common domain service to advertise the ability to authenticate a principal" is enabled (see *Configure standard IdP Discovery* on page 156).
- To create a connection for attribute query, select Attribute Query. See *Attribute Query and XASP* on page 42 in the "Supported Standards" chapter of the *Get started with PingFederate* guide.
- Click **Next**.

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 in the **Server Configuration** > **Server Settings** > **Metadata Lifetime** screen (see *Configure metadata lifetime* on page 141).

Although optional, it is recommended that you turn on email notifications for SAML metadata update events in the **Server Configuration** > **Server Settings** > **Runtime Notification** screen (see *Configure runtime notifications* on page 130).

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. In 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 in 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** workflow, the administrative console reminds you to select it from the **Existing URL Name** list. 3. (Optional) 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 in the **Import Certificate** screen, and then click **Next**. 5. In 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. (Optional) Clear the **Enable Automatic Reloading** check box to disable automatic update. 4. Click **Load Metadata**.

Metadata medium	Steps

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** workflow (see *Manage metadata URLs* on page 171).

5. Click **Next**.

Identify the SP

On the General Info screen, you provide a required unique identifier and display name for a connection, as well as optional contact information. In addition, on this screen you can set the level of transaction logging for this connection partner (see *Runtime transaction logging* on page 93).

Field Descriptions

Field	Description
Partner's Entity ID/ Audience/ Partner's Realm (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. For a WS-Federation connection, this is your partner's Realm. This ID may have been obtained out-of-band or via a metadata file if you are using a SAML protocol (see *Export metadata to an XML file* on page 101). For STS-only connections, 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 under Server Settings (see *Specify federation information* on page 137), enter a virtual server ID in this field and click **Add**. Enter additional virtual server IDs as needed (see *Multiple virtual server IDs* on page 81).
Base URL	The fully qualified hostname and port on which your partner's federation deployment runs (e.g., `https://sso.thespcompany.com:9031`). This entry is an optional convenience, allowing you to enter relative paths to specific endpoints, instead of full URLs, during the configuration process.
Company	The name of the partner company to which you are connecting.
Contact Name	The contact person at the partner company.
Contact Number	The phone number of the contact person at the partner company.
Contact Email	The email address for the contact person at the partner company.
Application Name	The name of the application, accessible through the IdP Adapter interface (IdpAuthenticationAdapterV2) in the PingFederate Java SDK. (For more information about the SDK, see *SDK developer's guide* on page 539.)
Application Icon URL	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 539.)
Logging Mode	The level of transaction logging applicable for this connection (see *Runtime transaction logging* on page 93).

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click**Manage All**, if needed, to see a full list of connections.
3. Click **General Info** under the SP Connection tab.

Configure Browser-based SSO

Browser-based SSO (also, Browser SSO) is another term for secure SSO, which relies on a user's web browser and HTTP to broker XML identity-federation messaging 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 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 *Supported standards* on page 28 chapter of the *Get started with PingFederate* guide). 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 PingTrust 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.

Note: The configuration screens represented in this chapter show “SAML 2.0” in their left corners, unless they are exclusive to WS-Federation or SAML 1.x setup requirements. When the SAML 2.0 screens are also applicable to SAML 1.x and/or WS-Federation connections, the SAML 2.0 representations and discussions also apply to the other protocols, unless otherwise indicated.

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 289). Also, other configuration tasks may remain to be configured for new or modified connections, depending on selected connection options (see *Choose SP connection options* on page 251).

Important: For new connections you must completely configure these SSO settings and subsequent tasks before you can save the connection on the Activation & Summary screen. Until then, the *configuration is temporary and can be lost*; the console times out after several minutes of inactivity. At any time, however, you can click **Save Draft**, which is available on most screens after you enter General Information (see *Console buttons* on page 27 in the “Console Navigation” chapter of the *Get started with PingFederate* guide).

Use the lists and links (or page references) below to find specific information about steps that may apply to your SSO connection requirements:

SAML 2.0 SSO Steps

- *Choose SAML 2.0 profiles* on page 255
- *Set an assertion lifetime* on page 257
- *Configure assertion creation* on page 257
  - *Choose an identity mapping method* on page 258
  - *Create an attribute contract* on page 259
  - *Manage authentication source mappings* on page 261
- *Configure protocol settings* on page 279
  - *Set Assertion Consumer Service URLs (SAML)* on page 279
  - *Specify SLO service URLs (SAML 2.0)* on page 282
  - *Choose allowable SAML bindings (SAML 2.0)* on page 283

- *Set an artifact lifetime (SAML)* on page 283
- *Specify artifact resolver locations (SAML 2.0)* on page 283
- *Define signature policy* on page 284
- *Specify XML encryption policy (for SAML 2.0)* on page 351

WS-Federation SSO Configuration Steps

- *Set an assertion lifetime* on page 257
- *Configure assertion creation* on page 257
- *Define signature policy* on page 284
  - *Choose an identity mapping method* on page 258
  - *Create an attribute contract* on page 259
  - *Manage authentication source mappings* on page 261
- *Configure protocol settings* on page 279
  - *Define a service URL (WS-Federation)* on page 281

SAML 1.x SSO Configuration Steps

- *Set an assertion lifetime* on page 257
- *Configure assertion creation* on page 257
  - *Choose an identity mapping method* on page 258
  - *Create an attribute contract* on page 259
  - *Manage authentication source mappings* on page 261
- *Configure protocol settings* on page 279
  - *Set Assertion Consumer Service URLs (SAML)* on page 279
  - *Set a default target URL (SAML 1.x)* on page 280
  - *Set an artifact lifetime (SAML)* on page 283
  - *Define signature policy* on page 284

Choose SAML 2.0 profiles

A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use (see *Federation planning checklist* on page 79). For SAML 2.0, PingFederate supports all IdP- and SP-initiated SSO and SLO profiles.

For information on typical SSO/SLO profile configurations, including illustrations, see *Browser-based SSO* on page 30in the *Get started with PingFederate* guide..

Note: This screen is not presented for SAML 1.x connections because IdP SSO is assumed, the SLO profiles are not supported, and the server supports SP-initiated SSO automatically (see *SAML 1.x profiles* on page 30 chapter of the *Get started with PingFederate* guide).

The screen is also not presented for WS-Federation connections because profile selection is not required (see *WS-Federation* on page 43 in the *Get started with PingFederate* guide).

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.
   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.

5. Click **SAML Profiles** on the Summary screen.

To configure profiles:

1. Select the profile(s) applicable to this connection and click **Next**.

   You must select an SSO profile before you can enable SLO.
2. Continue through the remaining connection-configuration tasks.

The following topics provide more information on requirements for each SAML profile:

- *About IdP-initiated SSO* on page 256
- *About SP-initiated SSO* on page 256
- *About IdP-initiated SLO* on page 256
- *About SP-initiated SLO* on page 257

About IdP-initiated SSO

When PingFederate is operating as an IdP, the IdP-initiated SSO profile configuration defines the message-transport mechanisms (bindings) your enterprise has agreed to use for this partner, plus optional digital signature requirements for outbound assertions (see *Security infrastructure* on page 72).

For illustrations of typical profile and binding scenarios, see the “Profiles” sections in the *Supported standards* on page 28 chapter of the *Get started with PingFederate* guide.

For this configuration you need to know:

- The transport binding(s) to which you and your partner have agreed
- The certificate to be used for signing assertions (not always required for the artifact binding)
- The URL(s) of your partner's Assertion Consumer Service(s)

About SP-initiated SSO

The SP-initiated profile configuration for SSO defines the message-transport mechanisms (bindings) and security requirements for receiving authentication requests and sending assertions when your SP partner initiates SSO transactions (see *Single Sign-on* in the “Supported Standards” chapter of the *Get started with PingFederate* guide).

For SAML 1.x, the SP-initiated SSO profile is also known as the “destination-first” profile, which was added as a supported “non-normative” use case after the release of the SAML 2.0 specifications. As an IdP, you do not need to configure this profile; when the SP sends an authentication request to your SSO Service endpoint, PingFederate will handle the response automatically.

For illustrations of typical profile and binding scenarios, see the Profiles in *Supported standards* on page 28 in the *Get started with PingFederate* guide.

For this configuration you will need to know:

- The endpoint URL(s) for your SP's Assertion Consumer Service(s)
- The transport bindings that you and your partner have agreed upon inbound and outbound
- The certificates you will use to sign outbound assertions and to verify incoming digital signatures from your SP, when either is required

When Artifact is an allowable inbound SAML binding, you also need to know the endpoint(s) to your partner's Artifact Resolution Service(s) and the SOAP client authentication mechanism to use: either HTTP Basic, SSL client certificates, a digital signature, or a combination of these mechanisms.

About IdP-initiated SLO

The SAML 2.0 IdP-initiated SLO profile configuration defines the message-transport mechanisms (bindings) and security requirements for exchanging SLO requests and responses.

For more information about SLO, see *Single logout* on page 41 in the “Supported Standards” chapter of the *Get started with PingFederate* guide.

For this configuration you need to know:

- The transport bindings to which you and your partner have agreed to send SLO requests and receive responses
- The certificates to be used for signing outgoing messages and for verifying incoming digital signatures from your SP (not always required for the artifact binding)
- The URL(s) of your SP's Single Logout Service(s)

About SP-initiated SLO

The SAML 2.0 SP-initiated SLO profile configuration defines the message-transport mechanisms (bindings) and security requirements that you and your partner have agreed upon for exchanging SAML requests and responses.

For more information about SLO, see *Single logout* on page 41 in the "Supported Standards" chapter of the *Get started with PingFederate* guide.

For this configuration you need to know:

- The transport bindings that you and your partner have agreed upon to send SLO requests and receive responses
- The certificates to be used for signing outgoing messages and for verifying incoming digital signatures from your SP (not always required for the artifact binding)
- The URL(s) of your SP's Single Logout Service(s)
- The URL of your SP's Artifact Resolution Service(s)—if this binding and endpoint are different from your SSO configuration—and SOAP client authentication requirements

Set an assertion lifetime

Identity-federation standards require a window of time during which an assertion is considered valid. Each assertion has a time-stamp XML element and elements indicating the allowable lifetime of the assertion (in minutes) before and after the assertion time stamp.

Field Descriptions

Field	Description
Minutes Before	The amount of time before the assertion was issued during which it is to be considered valid.
Minutes After	The amount of time after the assertion was issued during which it is to be considered valid.

To change the default times:

- Edit the desired setting(s) and click **Next** or **Save**.

  Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), specify values for Minutes Before and Minutes After in the JWT Lifetime screen.

Configure assertion creation

As an IdP, you must specify how PingFederate obtains user-authentication information and use it to create assertions 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 must associate one or more authentication policy contracts to the SP connection (see *Federation hub* on page 83 and *Deploying PingFederate as a federation hub* on page 86 for more information).

For both scenarios, the configuration includes:

- Choosing an identity-mapping method (see *Choose an identity mapping method* on page 258).
- Defining the attribute contract you will use with this partner, if any (see *Create an attribute contract* on page 259).
- Configuring instances of one or more authentication sources (see *Manage authentication source mappings* on page 261) and specifying how they are used to fulfill the attribute contract.
- Click **Configure Assertion Creation** to continue.

Note: The same steps apply to WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250) as well. Click **Configure JWT Creation** to continue.

Choose an identity mapping method

PingFederate allows your SP partner 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 66). At the Identity Mapping step, you choose the type of name identifier your partner needs to facilitate one of these options. You and your partner may want to decide in advance which option to use (see *Federation planning checklist* on page 79).

The choices of name-identifier types depend on which protocol you are using:

- For information about SAML selections, see *SAML Name ID selections* on page 258.
- For information about WS-Federation selections, see *WS-Federation Name ID selections* on page 259.

SAML Name ID selections

The allowable types of name identifiers for SAML connections are described below. These choices affect how SPs make use of account mapping or account linking.

If your SP 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 SP site (see *About attributes* on page 67).

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 send a known attribute to identify a user—for example, a username or email address—then select **Standard**.
- If you and your partner have agreed to use an opaque persistent name identifier (often used for account linking), then select **Pseudonym** on the Identity Mapping screen.

  The pseudonym is based on values pulled from the IdP adapter instance used to authenticate the user. You select these values when you configure IdP adapter instances (see *Set pseudonym and masking options* on page 240).

  To set up an attribute contract to use in conjunction with an opaque identifier, select the check box next to "Include attributes . . ." after selecting **Pseudonym**.

- 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 (see *Account linking* on page 67).

  A typical application for this selection might be, for example, when an SP provides generalized group accounts based on organizational rather than individual identity.

  To set up an attribute contract to use in conjunction with an opaque identifier, select the check box next to "Include attributes . . ." after selecting **Transient**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Identity Mapping** on the Summary screen.

WS-Federation Name ID selections

For WS-Federation purposes a name identifier is a uniquely identifying user attribute.

- Make one of the selections described below:
  - **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 example, if the SP has a group account set up for all users in a particular domain.

  Later, you will map your choice to the `SAML_SUBJECT` attribute in the SAML assertion (see *Map default attribute contract fulfillment* on page 277).

Create an attribute contract

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in SAML assertions for this connection (see *Attribute contracts* on page 68). You identify these attributes on this screen.

If you are sending a "standard" name identifier (see *Choose an identity mapping method* on page 258), then the contract includes the default `SAML_SUBJECT`, which identifies the user in the assertion. You will configure this variable later to contain a user ID or another agreed-upon attribute—for instance, an email address—that uniquely identifies the user (see *Configure attribute contract fulfillment* on page 273).

Creating an attribute contract is optional if you are sending either a pseudonym or a transient identifier to your connection partner (see *Choose an identity mapping method* on page 258).

Note: If you are configuring a WS-Federation connection using JWT (see *Choose an SP connection type* on page 250), you also need to create an attribute contract.

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.

   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, (see *Configure attribute contract fulfillment* on page 273).

   The specified value overrides the authentication context provided by the IdP adapter or the Requested AuthN Context Authentication Selector.

   Note that if no authentication context is provided by the `SAML_AUTHN_CTX` attribute, the IdP adapter, or the Requested AuthN Context Authentication Selector, 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.incommonfederation.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.

2. Optional: Select the Attribute Name Format.

   Note: This option is not available for SAML 1.0 connections or WS-Federation connections using JWT.

   Change the default Name Format selection if you and your SP partner have agreed to a specific format (see *Name formats* on page 68).

   Note: If needed, an administrator can customize name-format alternatives via the `custom-name-formats.xml` configuration file located in this directory:

   `<pf_install>/pingfederate/server/default/data/config-store`

3. Click **Add**.

(Optional) For SAML connections, to modify the Subject Name Format:

- Make your selection in the drop-down list and click **Done**.

  The default selection is usually applicable, unless you and your partner have agreed to a different format specification (see *Name formats* on page 68).

  Note: This selection is not available if you are sending a pseudonym as the subject, or a transient identifier (see *Choose an identity mapping method* on page 258).

To modify an attribute name or format:

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.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Attribute Contract** on the Summary screen.

   If this step is not in the list, then you have chosen to send either a pseudonym or a transient identifier without additional attributes (see *Choose an identity mapping method* on page 258).

   Note: For WS-Federation connections using SAML 1.1, the Subject Name Format is determined by the previous screen (see *WS-Federation Name ID selections* on page 259).

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Manage authentication source mappings

IdP adapters are responsible for handling user authentication as part of an SSO operation (see *SSO integration kits and adapters* on page 64). A configured and deployed adapter in PingFederate is known as an adapter instance. The same instance may be mapped by multiple connections.

You may also deploy an SP connection to bridge a service provider to one or more identity providers.

Tip: In this scenario, PingFederate is a federation hub for both parties. (See *Federation hub* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.)

PingFederate uses authentication policy contracts to associate this SP connection with the applicable IdP connections to the identity providers (see *Deploying PingFederate as a federation hub* on page 86).

Each authentication policy contract has its own set of attributes which you map values to the assertions.

Map one or more IdP adapter instances into each SP connection so that when a user authenticates with a particular external identity management system the user attributes are returned to PingFederate.

When adapters are restricted to certain virtual server IDs, the allowed IDs are displayed under the Virtual Server IDs column.

Regardless of how many IdP adapter instances are mapped in an SP connection, PingFederate uses only one instance to authenticate a user. Because each instance may return different user attributes, each IdP adapter mapping must define how the attribute contract is fulfilled; you must map attributes from each adapter—and/or attributes retrieved from your local data stores—into the assertions PingFederate sends to this SP to fulfill the attribute contract.

You begin this configuration on the Authentication Source Mapping screen, where you choose to map instances of IdP adapter or authentication policy contracts. If you have not yet configured an instance of the adapter you intend to use within this SP connection, see *Manage IdP adapters* on page 238. To review or create authentication policy contracts, see *Manage policy contracts* on page 235.

To modify an existing authentication source:

- Click its Name link.

To begin configuring an Adapter Instance for this connection

- Click **Map New Adapter Instance**.

To begin configuring an authentication policy contract for this connection:

- Click **Map New Authentication Policy**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Select an authentication source

Use this screen to associate an IdP adapter instance or an authentication policy contract with an SP connection.

Tip: An IdP adapter instance is available for use within an SP connection only after it has been deployed and configured in PingFederate.

To select an adapter instance:

- Choose an Adapter Instance from the drop-down list and click **Next** to continue.

  (Optional) To override any IdP adapter instance, select the **Override Instance Settings** check box (see *Override an IdP adapter instance* on page 263).

  If the adapter instance you need is not available, click **Manage Adapter Instances** to define one or more adapter instances you need for this connection.

  Note that an adapter instance can be mapped only once per connection. However, the same adapter instance may be mapped by multiple connections.

To select an authentication policy contract:

- Choose a Authentication Policy Contract from the drop-down list and click **Next** to continue.

  If the authentication policy contract you need is not available, click **Manage Authentication Policy Contracts** to define one or more authentication policy contracts you need for this connection.

  Note that an authentication policy contract can be mapped only once per connection. You may however map the same contract to multiple connections.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Adapter Instance** or **Authentication Policy Contract** on the Summary screen.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Override an IdP adapter instance

Overrides at the connection level simplify adapter management by allowing you to create a small set of adapter instances that define the base configuration requirements and then overriding settings at the connection level as needed.

You may override any IdP adapter settings at the connection level either during or after connection mapping.

Alternatively, you can override any adapter instance to apply to several connections, see *Hierarchical plug-in configurations* on page 66 for more information about adapter overrides.

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, make your changes, and then click **Next**. When you are finished, click **Done** to continue with IdP mapping.

  Note: Override adapter-setting screens are functionally identical to those used for creating a new adapter connection. Refer to the table below to find sections in this manual containing configuration information and procedures.

  The display of some screens listed in the table depends on the type of adapter you are configuring.

  For information about the override adapter-setting screens, depending on the type of setting, use the following table:

Override Screen	Manual Section
Adapter	See *Configure an IdP adapter instance* on page 239.
Actions	See *Invoke IdP adapter actions* on page 239.
Extended Contract	See *Extend an IdP adapter contract* on page 240.
Adapter Attributes	See *Set pseudonym and masking options* on page 240.

- To remove any adapter connection override, clear the **Override Instance Settings** check box, and then complete the rest of the setup.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Adapter Instance** on the Summary screen.
10. Select the **Override Instance Settings** check box.
11. Click **Next**.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

View the IdP adapter type

Adapter override instance type information.

This screen is view only. This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Configure an IdP adapter instance* on page 239.

Override IdP adapter settings

- If you want to override any settings on this screen, select the override check box related to the field(s) you want to modify, make your changes, and then click Next.

  This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Configure an IdP adapter instance* on page 239.

Override the IdP adapter extended contract

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Extend an IdP adapter contract* on page 240.

Override the IdP adapter actions

This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Invoke IdP adapter actions* on page 239.

Override the IdP adapter attributes

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Set pseudonym and masking options* on page 240.

Review the overridden IdP adapter instance

- On the Override Instance Settings Summary screen, click any heading to change your adapter configuration; or click **Done** to continue.

Restrict an authentication source to certain virtual server IDs

When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), you have the option to enforce authentication requirements by restricting an authentication source (an adapter or an authentication policy contract) to certain virtual server IDs. This optional setting can be applied to each authentication source added to the connection using virtual server IDs. By default, no restriction is imposed.

To restrict an authentication source to a subset of the available virtual server IDs:

1. Select the **Restrict Virtual Server IDs** check box.
2. In the **Allowed Virtual Server IDs** area, select virtual server IDs that you want to allow for this authentication source.
3. Click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.

8. Click the authentication source name.
9. Click **Virtual Server IDs** on the Summary screen.

Note: The Virtual Server IDs screen is only available for connections using at least one virtual server ID, see *Federation Server Identification*.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Select an attribute mapping method

You can query local user-data stores to help fulfill the attribute contract, in conjunction with attribute values supplied by the authentication source (an IdP adapter or an authentication policy contract).

When using data stores, you can use one or more data stores to look up attributes for a single mapping. Alternatively, you can define alternate data stores and a failsafe mapping. The former and the latter are represented by the following check boxes in the **Mapping Method** screen:

- **Retrieve additional attributes from multiple data stores using one mapping**
- **Retrieve additional attributes from a data store — includes options to use alternate data stores and/or a failsafe mapping**

If you use only the Adapter Contract or Authentication Policy Contract values, then you map values for the attribute contract next (see *Map default attribute contract fulfillment* on page 277).

If you choose to retrieve additional attributes, then you identify data stores and specify lookup queries next (see *Configure attribute sources and user lookup* on page 265).

Tip: To determine whether you need to look up additional values, compare the attribute contract against the adapter contract or the authentication policy contract (see *Create an attribute contract* on page 259 and *Manage authentication source mappings* on page 261). If the attribute contract requires more information, determine whether local data stores can supply it. (You can also choose to use text constants or expressions for certain information—see *Map default attribute contract fulfillment* on page 277.)

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Mapping Method** on the Summary screen.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

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 (see *Create an attribute contract* on page 259). Attribute sources can be reused across connections to other SP partners.

You can use more than one attribute source when mapping values to the attribute contract:

- Set up search parameters for your data stores, including “fall-through” searches. If any search fails, the next search executes until all searches are exhausted, at which point a configured failsafe mapping executes. If the failsafe mapping is not configured, the SSO transaction fails.

Note: Queries are executed in the order of Attribute Sources shown. Use the up/down arrows as needed to adjust the order. Note, however, that data can originate from only one source.

- Configure separate data stores to look up attributes for a single mapping. If any search fails to find a user, then the SSO transaction fails.

To configure an attribute source:

- Click **Add Attribute Source** and complete the setup steps (see *Set up an attribute source* on page 266 next).

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:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup**.

   If this step is not listed, then this instance is configured to use values from the authentication source (an adapter or an authentication policy contract) only (see *Select an attribute mapping method* on page 265).

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Set up an attribute source

For attribute-source setup information, refer to the sections indicated in the following steps.

Note: As you make selections on configuration screens, ensure that you allow enough time for PingFederate to access your data store and populate drop-down lists.

1. See *Select a data store* on page 267.
2. See the following sections in this manual, depending on the type of data store:

Data Store Type	Related Manual Section
JDBC	• *Select a database table and columns* on page 268 • *Specify a database filter* on page 269
LDAP	• *Configure an LDAP search* on page 270 • *Specify an LDAP filter* on page 272
Custom	• *Configure custom source filters* on page 273 • *Select custom source fields* on page 273

3. See *Configure attribute contract fulfillment* on page 273.

Data store selection

Data-store configuration is the same for all attribute source and user lookup task flows. The Data Store screen allows you to define data stores to look up attributes. The values extracted are used, for example, to help fulfill attribute contracts, attribute requests, and adapter contracts.

For the appropriate attribute-source setup steps, refer to one of the following help topics:

- IdP Adapter Contract Mapping (see *Define the IdP adapter contract* on page 240)
- Browser SSO (SP Connection) IdP Adapter Mapping (see *Select a data store* on page 267)
- Attribute Query (SP Connection) User Attribute Mapping (see *Choosing a User-Data Store*)
- STS (SP Connection) IdP Token Processor Mapping (see *Configure a data store for token creation* on page 393)
- IdP Adapter to SP Adapter Mapping (see *Choosing a Data Store (optional)*)

Select a data store

This screen allows you to choose a data store from a previously configured list (see *Manage data stores* on page 143). Attribute values extracted from this data store are used to help fulfill the attribute contract or the authentication policy contract contract for this partner (see *Create an attribute contract* on page 259).

To define an attribute source:

1. Enter an Attribute Source Id to uniquely identify the data source for the mapping.

   Note: This field appears only if you are retrieving additional attributes from multiple data stores using one mapping (see *Select an attribute mapping method* on page 265).
2. Use Attribute Source Description to specify an attribute source name that distinguishes this user lookup for the selected data store.

   Note: When using multiple data stores for one mapping, PingFederate appends this description to the data store type in the Source list on the Attribute Contract Fulfillment screen (see *Configure attribute contract fulfillment* on page 273).
3. Choose an Active Data Store and click **Next**.

   A data-store configuration must be defined under System Settings for use within a connection. If the data store you want is not shown in the drop-down menu, click **Manage Data Stores** to add it (see *Manage data stores* on page 143).

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Select a database table and columns

When you choose to use a database source for attributes, you follow this path through the configuration steps.

On this screen you begin to specify exactly where additional data can be found to complete the attribute contract when you send an assertion to this SP (see *Create an attribute contract* on page 259). Only one table may be used as a source of data for a JDBC lookup.

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 *http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html*.

Field descriptions

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 a JDBC query. Other databases, such as MySQL, do not require selection of a schema.
Table	Displays the table(s) 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 query.

To select a database table and columns for queries:

1. Choose a Schema file (when applicable) from the drop-down list.
2. Choose a Table from the drop-down list.
3. Choose a name under Columns to Return from Select and click **Add Attribute**.

   Tip: Click **Refresh** if you are updating an existing configuration and changes may have been made to the database.

   Repeat this step for other columns as needed.

   Note: You do not need to add a column here for it to be used as part of a search filter (see *Specify a database filter* on page 269 next). Add only attributes from which you need actual values to pass to the SP.

   Tip: To determine what attributes to look up during a query, click the **View Attribute Contract** link to see what information must be collected (see *Create an attribute contract* on page 259). Then determine what information is coming in from the session lookup adapter (see *Select an attribute mapping method* on page 265). Information not contained in the authentication source (an adapter contract or an authentication policy contract) may be pulled from the data store look-up query.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.

5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.
11. Click **Database Table and Columns**.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Specify a database filter

The JDBC `WHERE` clause in PingFederate queries the data table you selected to retrieve a record associated with a particular value (or values) from the assertion. The clause is in the form:

```
WHERE column1=value1 [AND column2=value2] [O...]
```

The left side of the first variable pair uses a column name in the database table you selected (see *Select a database table and columns* on page 268).

The right side generally uses values passed in from your authentication source (an adapter or an authentication policy contract).

Note: 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. For more information on multiple data-store mapping, see *Multiple data source attribute mapping* on page 70.

You can also apply additional search criteria from your own database, using any other columns from the targeted table.

Tip: Click "**View List of Columns . . .**" to see a list from which to copy and paste.

For more information about `WHERE` clauses, consult your DBMS documentation.

EXAMPLE:

```
userid='${username}'
```

In this example `userid` is the name of a column in the JDBC data store. On the right side, `'${username}'` returns the value of the `username` variable from the IdP adapter.

Important: You *must* use the `${}` syntax to retrieve the value of the enclosed variable and use single quotation marks around the `${}` characters.

Field description

Field	Description
Where	`WHERE` clause statements conditionally select data from a table. Enter the `WHERE` clause statement in the space provided. For example: `WHERE email='clive@company.com'`.

To construct the `WHERE` clause:

1. Enter the statement in the space provided, following the guidelines and example above.

   The initial `WHERE` is optional.
2. Ensure the syntax and variable names are correct.

When you click **Next**, you will map attribute values returned from the database into the attribute contract (see *Configure attribute contract fulfillment* on page 273).

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
8. Click the attribute source Description link.
9. Click **Database Filter** from the steps list.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Configure an LDAP search

When you choose to use an LDAP source for attributes, you follow this path through the configuration steps.

On this screen you specify the branch of your LDAP hierarchy where you want PingFederate to look up user data.

Field descriptions

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	Determines the node depth of the query. Select Subtree, One level or Object.
Root Object Class	The class containing the attributes you want.
Attributes to return from search	A list of attributes added from the drop-down list below. Subject DN is a default attribute, which may be used as the primary user identifier.

To select LDAP attributes:

1. Optional: Enter a Base DN.
2. Select a Search Scope.
3. Select a Root Object Class.
4. Under Attributes to return from search, choose an attribute and click **Add Attribute**.

   Note that the attribute Subject DN is always returned by default.

   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 as well as indirectly through nested group membership (if any) under the Base DN.

   For example, suppose you have three groups under the Base DN, namely 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).

5. Repeat the last step for other attributes as needed.

Note: You do not need to add an attribute here for it to be used in a search filter (see *Specify an LDAP filter* on page 272). Add only attributes from which you need actual values to pass to the SP.

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.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.
11. Click **LDAP Directory Search** from the steps list or fill out the appropriate screens and advance to this screen.

    If you have not yet defined an LDAP data store, see *Select a data store* on page 267.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Specify 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. Because binary attribute data cannot be used in an assertion, use this screen to specify which encoding type (Base64, Hex or SID) you want to apply during attribute contract fulfillment.

For example, Microsoft Office 365 uses objectGUID, an immutable Active Directory binary attribute associated with user accounts. Office 365 requires this binary data to be Base64-encoded to correlate provisioned federated user data to Active Directory accounts.

Claims-based authentication with Microsoft Outlook Web App and Exchange admin center (EAC) is another example. It requires tokenGroups (another binary attribute in Active Directory) to be SID-encoded.

Note: Attributes are specified as binary when configuring an LDAP connection (see *Specify LDAP binary attributes* on page 148).

To select an attribute encoding type:

- Select the type of attribute encoding you want to apply for each attribute and click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.
11. Click **LDAP Binary Attribute Encoding Types** from the steps list.

Specify an LDAP filter

The LDAP filter queries 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:

```
(attribute=${value}).
```

The left-side variable is an attribute you selected earlier (see *Configure an LDAP search* on page 270).

The right side generally uses values passed in from your authentication source (an adapter or an authentication policy contract).

Note: 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. For more information on multiple data-store mapping, see *Multiple data source attribute mapping* on page 70.

You can also apply additional search criteria from your data store, using any other attributes from the targeted object classes.

Tip: Click "**View List of Available LDAP Attributes**" for a list from which you can copy and paste.

For general information about search filters, consult your LDAP documentation.

Field descriptions

Field	Description
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: 1) at least one attribute (attribute data type) to search on, 2) a search filter operator that will determine what to match, and 3) the value of the attribute being sought. Searches must have at least one of each of these three components.

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.

   Note: If you used an anonymous binding to create this LDAP connection, your access might be restricted (see *Configure an LDAP connection* on page 145).
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.
11. Click **LDAP Filter** from the steps list.

    If you have not yet defined an LDAP data store, see *Select a data store* on page 267.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Configure custom source filters

When you choose to use a custom source for attributes, you follow this path through the configuration steps.

On this screen you specify a filter, or lookup 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.

Select custom source fields

On the Configure Custom Source Fields screen, you can choose from among the fields shown to map to the attribute contract. These choices are supplied by the driver implementation. Select only those needed to fulfill the attribute contract for this partner connection.

Configure attribute contract fulfillment

The last step in configuring an attribute source is to map values into the attribute contract (see *Create an attribute contract* on page 259). These are the values included in assertions sent to this SP (provided the information is found in this attribute source).

You map attributes on the Attribute Contract Fulfillment screen.

Map each attribute to fulfill the Attribute Contract from one of these Sources:

- Adapter or Authentication Policy Contract (the authentication source)

  Values are returned from the authentication source. When you make this selection, the associated Value drop-down list is populated by the authentication source (an adapter or an authentication policy contract).

  For example, you might choose the adapter attribute username to map to `SAML_SUBJECT`.
- Context

  Values are returned from the context of the transaction at runtime.

  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—the service provider should take the information about the original issuer into consideration before granting access to protected resources.

  See *Federation hub* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.

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 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)

- LDAP/JDBC/Custom

Note: PingFederate appends a description in parentheses for data stores of the same type (see *Select a data store* on page 267).

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 (see *Configure an LDAP search* on page 270, *Select a database table and columns* on page 268, or *Configure custom source filters* on page 273).

- 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 (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 authentication source (an adapter or an authentication policy contract), 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 *Select a data store* on page 267) and `attribute` is any of the data store attributes you select.

When using alternate data stores and/or a failsafe mapping, the Attribute Source Id field is not presented. Use the following syntax in this instance:

`${ds.attribute}`

where *`attribute`* is any of the data store attributes you select.

Tip: Two other 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; this variable is available only if specified as a query parameter for the relevant PingFederate endpoint (either as `TargetResource` for SAML 2.0 or `TARGET` for SAML 1.x—see *Application endpoints* on page 442).

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.

To map attributes:

1. Choose a Source for each Target attribute.

   See *Map each attribute to fulfill the Attribute Contract from one of these Sources:* above.
2. Choose (or enter) a Value for each Attribute.

   All values must be mapped.
3. Click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.

2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Sources & User Lookup** under the IdP Adapter Mapping or Authentication Policy Contract tab.

   If this step is not shown, you have elected not to look up attributes in data stores (see *Select an attribute mapping method* on page 265).
10. Click the attribute source Description link.
11. Click **Attribute Contract Fulfillment** from the steps list.

    (If you have not yet defined a data store, see *Select a data store* on page 267).

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Specify issuance criteria (optional)

Use this screen to define criteria PingFederate can evaluate to determine whether to issue an assertion for a user (see *About token authorization* on page 71). This token authorization can be used to restrict who can SSO to protected resources.

Note: This screen does not appear if you chose to use data stores with the failsafe mapping option (see *Select an attribute mapping method* on page 265).

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Adapter or Authentication Policy Contract – Select to access attributes from the authentication source.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.
   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for data stores of the same type (see *Select a data store* on page 267)
   - Mapped Attributes – Select to access the required attributes.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

Error results are handled differently for SP-initiated and IdP-initiated SSO:

SP-Intiated SSO

- **SAML** – The Error Result field is used by the `StatusMessage` element in the response to the SP.
- **Template (WS-Federation)** – The Error Result field is used by the variable `$errorDetail in the sourceid-wsfed-idp-exception-template.html` template (for more information on this and other user-facing templates, see *Customizable user-facing screens* on page 120).

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 (for more information on this and other user-facing templates, see *Customizable user-facing screens* on page 120).

Note: Using an error code in the Error Result field allows the error template or a web application to process the error result in a variety of ways—for example, a redirect to another page or e-mail to an administrator.

If you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Enable and disable expressions* on page 485).

   Important: When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), consider using an OGNL expression to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access (see *Sample OGNL expressions* on page 487).

   - Use the in-line editor box to enter the OGNL expression.

     For more information about OGNL, see *Attribute mapping expressions* on page 485.
   - Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.
   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

     Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

Review the attribute source summary

When you have finished configuring Attribute Sources and User Lookup, 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** to continue with the rest of the configuration. If you are editing an existing connection, click **Done** on successive screens until you reach the Assertion Creation screen and then click **Save**.

Specify a failsafe attribute source

When a data store is configured and the attribute mappings under Attribute Sources & User Lookup fail to complete the attribute contract, 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.) The failsafe mapping is used instead of the mapping configured with the data-store setup.

Note: This screen does not appear if you chose to retrieve attributes from multiple data stores using a single mapping (see *Select an attribute mapping method* on page 265).

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.

Note: If you chose to have PingFederate stop the SSO transaction, the Attribute Contract Fulfillment screen is not presented.

To specify whether to use a failsafe attribute source:

- Make the relevant selection to use either a default set of attributes or to terminate the SSO, and then click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.
7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Failsafe Attribute Source** on the Summary screen.

   This step appears only if you chose to use alternate data stores and/or a failsafe mapping when retrieving attributes (see *Select an attribute mapping method* on page 265).

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Map default attribute contract fulfillment

Fulfillment of the attribute contract must be specified whether or not data sources are used. You accomplish this on the Attribute Contract Fulfillment screen, either by choosing to configure default mappings after setting up attribute sources (see *Configure attribute contract fulfillment* on page 273) or if you choose not to set up attribute sources (see *Select an attribute mapping method* on page 265).

Map each attribute to fulfill the Attribute Contract from one of these Sources:

- Adapter or Authentication Policy Contract (the authentication source)

  Values are returned from the authentication source. When you make this selection, the associated Value drop-down list is populated by the authentication source (an adapter or an authentication policy contract).

  For example, you might choose the adapter attribute username to map to `SAML_SUBJECT`.
- Context

  Values are returned from the context of the transaction at runtime.

  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—the service provider should take the information about the original issuer into consideration before granting access to protected resources.

  See *Federation hub* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.

  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 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)
- 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 (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 authentication source (an adapter or an authentication policy contract), using the `${attribute}` syntax.

  Tip: Two other 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; this variable is available only if specified as a query parameter for the relevant PingFederate endpoint (either as `TargetResource` for SAML 2.0 or TARGET for SAML 1.x—see *Application endpoints* on page 442).

To map attributes:

1. Choose a Source for each Target attribute (see descriptions of each Source under *Map each attribute to fulfill the Attribute Contract from one of these Sources:* above).
2. Choose (or enter) a Value for each Attribute.

   All values must be mapped.
3. Click **Next**.

To reach this screen:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Assertion Creation** under the Browser SSO tab.
6. Click **Configure Assertion Creation**.

7. Click **Authentication Source Mapping** on the Summary screen.
8. Click the authentication source name.
9. Click **Attribute Contract Fulfillment** on the Summary screen.

   If you are using data stores for attribute mapping and this step does not appear, see *Specify a failsafe attribute source* on page 277.

Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), click **JWT Creation** and **Configure JWT Creation** in steps 4 and 5, respectively.

Review the authentication source mapping

When you have finished adding a new or modifying an existing instance of an IdP Adapter or an 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.

- If you are editing an existing configuration, click **Done**; if you want to keep your changes, click **Save** when you reach the Browser SSO screen.

Review the assertion creation summary

- On the Assertion Creation Summary screen, click any heading to change the configuration; or click **Done** to continue.

  If you are editing an existing configuration, be sure to click Save if you are finished.

  Note: The same steps apply to WS-Federation connections using JWT (see *Choose an SP connection type* on page 250) in the JWT Creation Summary screen.

Configure protocol settings

The Protocol Settings screen provides the launching point for configuring bindings, partner endpoints, and other settings needed for the selected SAML profiles (if you are using SAML 2.0—see *Choose SAML 2.0 profiles* on page 255). The screen also displays configured information.

(For WS-Federation, the configuration of bindings is not applicable.)

To configure Protocol Settings, you need to know:

- For SSO profiles, the URL(s) of your SP's Assertion Consumer Service(s)
- For SLO profiles, the URL(s) of your SP's Single Logout Service(s)
- When artifact is an allowable inbound binding, the URL of your SP's Artifact Resolution Service(s)
- The transport configurations (bindings) that you will use to send and receive data for SSO/SLO connections
- Digital signature policies and certification requirements to which you and your connection partner have agreed
- XML encryption policies to which you and your connection partner have agreed

Set Assertion Consumer Service URLs (SAML)

At this step for SAML connections, you associate bindings to the Assertion Consumer Service (ACS) endpoint(s) where your SP will receive assertions. This configuration applies to either SSO Profile (see *Choose SAML 2.0 profiles* on page 255).

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.

Field description

Field	Description
Default (SAML 2.0)	A check in this check box indicates that the URL configuration in that row will be used as a default.
Index (SAML 2.0)	Uniquely identifies multiple ACS endpoints.
Binding	The method of transmission: POST or Artifact.
Endpoint URL	A location to which the assertion is sent, according to partner requirements.
MessageType	The type of the message that you need to customize for this connection.
Expression	The OGNL expression to fulfill your use case.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Assertion Consumer Service URL** on the Summary screen.

To define an Endpoint URL:

1. Select the Binding your partner specifies for the Endpoint.
2. Optional: Enter an Index value — 0 or 1, for example.

   For SAML 2.0 the specifications provide for the use of index numbers to identify multiple ACS endpoints. PingFederate supplies this number automatically; however, you can manually set the number to match your partner's configuration as needed.
3. Enter the fully qualified Endpoint URL or just a relative path if you have defined a base URL (see *Identify the SP* on page 253).
4. For SAML 2.0 connections, if this is the default (or only) endpoint, select the check box under Default.
5. Click **Add**.

To customize the SAML assertions:

1. Click **Show Advanced Customizations**.

   Note: Expressions must be enabled for the **Show Advanced Customizations** button to appear (see *Enable and disable expressions* on page 485).
2. Select a Message Type.
3. 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 489.

Set a default target URL (SAML 1.x)

This URL is used whenever PingFederate receives an SSO request from a local application that does not include the user's target resource URL at the SP site. The URL is required regardless of whether you expect your local application(s) to specify the target—to ensure that the server functions correctly during SSO events.

Field descriptions

Field	Description
Default Target URL	The URL of the target SP resource.

Define a service URL (WS-Federation)

The Service URL is the WS-Federation endpoint of your SP partner where you send SAML assertions and SLO cleanup messages. The assertions 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 WS-Federation SP partners when 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 in 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 159).

Some federation use cases may require additional customizations in the assertions sent from the PingFederate IdP server to the SP. You can use OGNL expressions to fulfill these use cases.

To define a service URL:

- Enter the fully qualified URL or just the relative path if you have defined a base URL (see *Identify the SP* on page 253). You must include the initial slash if you are entering only a relative path.

To specify additional allowed domains and paths:

1. Indicate whether to 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.

2. Enter the expected domain or IP address under Valid Domain Name.

   Use the domain only, without qualifiers. For example:

   `company.com`

   Using an initial wildcard and period for a domain name will cover multiple subdomains. For example: `*.company.com`

   covers `hr.company.com` or `email.company.com`.

   (Optional) Enter the exact path of the resource (case-sensitive) 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. For example:

   `/app/Consumer.jsp`

   allows `/app/Consumer.jsp` but rejects `/app/consumer.jsp`.

   Tip: You can also enter multiple query parameters with or without a fragment.

   For example: `/app/Consumer.jsp?area=West&team=IT#ref1001`

   matches `/app/Consumer.jsp?area=West&team=IT#ref1001` but not `/app/Consumer.jsp?area=East&team=IT#ref1001`

   (Optional) Select the Allow Any Query / Fragment check box if you want to allow any query parameters or fragment in the resource.

   Note: When selected, no query parameter and/or fragment is allowed in the path.

   Click **Add**.

3. Repeat the previous step to add additional entries as needed.
4. Click **Save**.

To customize the assertions:

1. Click **Show Advanced Customizations**.

   Note: Expressions must be enabled for the **Show Advanced Customizations** button to appear (see *Enable and disable expressions* on page 485).
2. Select a Message Type.
3. 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 489.

Specify SLO service URLs (SAML 2.0)

At this step you associate bindings to the endpoints where your SP receives logout requests when SLO is initiated at your site and where you send SLO responses when you receive SLO requests from the SP.

This step applies only to SAML 2.0 connections when you select either SLO profile (see *About IdP-initiated SLO* on page 256 or *About SP-initiated SLO* on page 257).

Field descriptions

Field	Description
Binding	The method of transmission: POST, Artifact, Redirect, or SOAP.
Endpoint URL	A location to which SLO logout request messages are sent, according to SP requirements.
Response URL	(Optional) A location to which SLO logout response messages are sent according to SP requirements. When omitted, the PingFederate IdP server sends logout responses to the Endpoint URL.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **SLO Service** URLs on the Summary screen.

To add a URL:

1. Select the Binding type.
2. Enter the fully qualified URL (or the relative path, if you have specified a base URL—see *Identify the SP* on page 253).
3. Optional: Enter the Response URL.
4. Click **Add**.

To edit an endpoint:

1. Click **Edit** under Action for the endpoint.
2. Make your change and click **Update**.

To delete an entry:

- Click **Delete** under Action for the endpoint.

Choose allowable SAML bindings (SAML 2.0)

At this step for SAML 2.0 connections, you select the binding(s) that your SP partner will use to send SAML authentication requests or SLO messages.

This configuration applies to SP-initiated SSO and to either SLO profile.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Allowable SAML Bindings** on the Summary screen.

Set an artifact lifetime (SAML)

When you send an artifact to your SP's Assertion Consumer Service or SLO service (for SAML 2.0), an element in the message indicates how long it should be considered valid.

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.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Artifact Lifetime** on the Summary screen.

   This step appears only if you have selected the artifact binding for either an SSO or SLO Service (under SAML 2.0) at the SP site.

Specify artifact resolver locations (SAML 2.0)

This endpoint or group of endpoints is where your server will send back-channel requests to resolve artifacts received from your partner. The locations are also known collectively under SAML specifications as the Artifact Resolution Service.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Artifact Resolver Locations** on the Summary screen.

If this step does not appear, you do not have Artifact selected under **Allowable SAML Bindings**.

To configure the Artifact Resolver Location(s):

1. Enter a URL on the Artifact Resolver Locations screen and click **Add**.

   The URL must be fully qualified (defining protocol, host, and port) unless you have entered a base URL (see *Identify the SP* on page 253).

   Repeat this step if your SP supports multiple services. The SAML 2.0 specifications permit multiple artifact resolution services through the use of Index numbers, which PingFederate automatically supplies when you add a service. Alternatively, if needed per partner specifications, you may assign these index numbers manually.

   Note: When specifying multiple artifact resolution endpoints, each endpoint must share the same transport protocol. That is, if one endpoint uses HTTP, then all must use HTTP. Similarly, if one endpoint uses HTTPS, then all must use HTTPS.

2. Click **Next**.

Define signature policy

The Signature Policy screen provides options controlling how digital signatures are used for SSO Internet messaging. The choices made on this screen depend on your partner agreement (see *Digital signing policy coordination* on page 74).

Digital signing is required for SAML Response messages sent from your site via POST (or Redirect for SAML 2.0). Optionally, SSO authentication requests from the SP (SP-initiated SSO) may also be signed to enforce security. (This option appears only for SAML 2.0 connections and only if you have enabled SP-initiated SSO using the POST or redirect bindings.)

The assertions inside SAML Responses may be also be signed. When you make this choice, only the assertion portion of the Response is signed, not the complete Response. (This is the only option that appears for SAML 1.x connections.)

- Make your selection(s) and click **Next**, or just click **Next** if no additional security is required.

Configure 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. This feature is commonly used if the assertion might pass through an intermediary (such as a user's browser) and HTTPS is not used.

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 single-logout 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 reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the SP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** under the Browser SSO tab.
6. Click **Configure Protocol Settings**.
7. Click **Encryption Policy** on the Summary screen.

To define XML encryption:

1. Choose whether you want to encrypt the entire assertion or one or more attributes.

2. If you are encrypting the name-identifier attribute, use the check boxes near the bottom of the screen to indicate whether you will also encrypt this attribute in outbound SLO messages and/or allow its encryption for inbound messages.
3. Click **Next** or **Done**.

To disable previously configured XML encryption selections:

1. Select **None** and then **Done**.
2. Click **Save** on the Protocol Settings screen.

Review protocol settings

On the Summary screen, you can review or edit your Protocol Settings.

Important: When you finish editing existing settings, be sure to click **Done** on the Summary screen and then **Save** on the Protocol Settings screen. For a new connection, click **Done** and then click **Next** on the Protocol Settings screen. Save the entire connection on the Activation & Summary screen (see *Review an SP connection settings* on page 304).

To reconfigure saved settings:

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 dependent or other changes, do so and continue by clicking **Done** until you reach the Protocol Settings screen.
3. Click **Save** on the Protocol Settings screen.

Review browser-based SSO settings

On the Summary screen for Browser SSO, you can review or edit your SSO configuration.

Important: When you finish editing existing settings, be sure to click **Done** on the Summary screen and then **Save** on the Browser SSO screen. For a new connection, click **Done** and then click **Next** on the Browser SSO screen. Save the entire connection on the Activation & Summary screen (see *Review an SP connection settings* on page 304).

To reconfigure saved settings:

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 dependent or other changes, do so and continue by clicking **Done** until you reach the Browser SSO screen.
3. Click **Save** on the Browser SSO screen.

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 251). 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* in the "Supported Standards" chapter of the *Get started with PingFederate* guide).

- 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 286).

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 **IdP Configuration** on the Main Menu.
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 an ID and a description 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 496 • *Enter a database search filter* on page 496
LDAP	• *Specify an LDAP directory and attributes* on page 497 • *Define encoding for LDAP binary attributes* on page 498 (optional)

Data store type	Required tasks
Custom	• *Enter an LDAP search filter* on page 499 • *Specify a custom source filter and fields* on page 499

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* in the "Supported Standards" chapter of the *Get started with PingFederate* guide.

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

  Values are returned from the context of the transaction at runtime.

  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—the service provider should take the information about the original issuer into consideration before granting access to protected resources.

  See *Federation hub* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.

  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 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)

- 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 *Choosing a User-Data Store*).

- Expression (when enabled)

This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 485). 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 *Choosing a User-Data Store*) 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.

Specify issuance criteria for attribute query (optional)

Use this screen to define criteria PingFederate can evaluate to determine whether to issue an attribute query response (see *About token authorization* on page 71). This extension of token authorization can be used to restrict who can access protected resources.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values returned from the context of the transaction at runtime.

     Note: For the HTTP Request selection, because it is retrieved as an object rather than text, you must generally use OGNL expressions for evaluation. Click **Show Advanced Criteria** to use expression mapping. (If that button is not displayed on the screen, expressions are not enabled—see *Enable and disable expressions* on page 485). For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)
   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source

     Note: PingFederate appends a description in parentheses for each data store lookup (see *Choosing a User-Data Store*).
   - Mapped Attributes – Select to access retrievable attributes defined for the attribute query profile.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   The Error Result field is used by the `StatusMessage` element in the SAML response to the SP.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the Show Advanced Criteria button to appear (see *Enable and disable expressions* on page 485).

   - Use the in-line editor box to enter the OGNL expression.

     For more information about OGNL, see *Attribute mapping expressions* on page 485.

   - Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information)

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

     Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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 289).

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 presents a list of possible 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* on page 290
- *Configure digital signature settings* on page 291
- *Configure signature verification settings* on page 292
- *Select an encryption certificate (SAML 2.0)* on page 294
- *Select a decryption key (SAML 2.0)* on page 295

- To continue, click **Configure Credentials**.

Configure back-channel authentication

When you configure a profile for inbound SAML messages via the artifact binding, you must specify authentication information for outbound artifact resolution requests over SOAP to your SP's Artifact Resolution Service.

Similarly, if you configure outbound Assertion Consumer Service or SLO Service URLs to use the artifact binding, then you must configure SOAP authentication requirements for inbound messages such as artifact resolution requests. If you configure outbound SLO Service URLs to use the SOAP binding, then you must also configure authentication requirements for outbound SOAP messages.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Back-Channel Authentication** on the Summary screen.

   If this step is not present, then it is not applicable to your configuration—you have not configured any profiles that use an artifact or SOAP binding or allowed artifact as an inbound SAML binding.

To configure back-channel authentication requirements for sending SOAP messages:

1. On the Back-Channel Authentication screen, click the **Configure** link to the right of the list of messages to be *sent* to your partner.
2. Make one or more selections on the Outbound SOAP Authentication Type screen:
   - HTTP Basic — you enter SOAP Basic credentials on a later screen.
   - SSL Client Certificate — you specify the certificate on a later screen.

     This option is available only if you specify an endpoint that uses SSL.
   - Use Digital Signatures (Browser SSO profile only) — you sign the message.

     You are asked to select a signing certificate on a later screen.

   For SAML 2.0, these options may be used in any combination or independently. For SAML 1.x, you must use either Basic or SSL authentication; digital signing may be added to ensure message integrity.

   By default, PingFederate validates your partner's SSL server certificate— verifying that the certificate chain is rooted by a trusted Certificate Authority and that the hostname matches the certificate's Common Name. Clear the associated check box if you do not want this validation to occur.
3. Click **Next**.
4. If you chose HTTP Basic at *Step 2*, enter the SOAP Username and Password to use for this partner under Basic SOAP Authentication.

   You must obtain these credentials from your partner.
5. If you are using an SSL certificate, select the certificate under SSL Authentication Certificate and click **Next**.

   If you have not yet created or imported the client SSL certificate you need into PingFederate, click **Manage Certificates** (see *Manage SSL client keys and certificates* on page 163). You need to export the certificate (only) and send it your partner.
6. On the Summary screen, click **Done**.

To configure back-channel authentication requirements for receiving SOAP messages:

1. On the Back-Channel Authentication screen, click the **Configure** link to the right of the list of messages to be *received* from your partner.
2. Select one or more options on the Inbound Authentication Type screen:

- HTTP Basic — Enter the logon username and password your partner uses on the next screen.
- SSL Certificate — Specify certificate verification information on a later screen.
- Use Digital Signatures (Browser SSO profile only). . . — Incoming messages must be signed.
- Require SSL — When selected, incoming HTTP transmissions must use a secure channel.

  You are asked to select a signature verification certificate on a later screen.

  For SAML 2.0, use these options in any combination or independently. For SAML 1.x, you must use either Basic or SSL authentication; add digital signing to ensure message integrity.

3. Click **Next**.
4. If you chose HTTP Basic at *Step 2*, enter the Username and Password under Basic Authentication (Inbound).

   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.
5. If you are using an SSL certificate, select Anchored or Unanchored under Certificate Verification Method.

   - Anchored — The certificate must be signed by a trusted Certificate Authority. 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 (see *Manage trusted certificate authorities* on page 160).
   - Unanchored — The 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.
6. Click **Next**.
7. If you chose anchored SSL certificate verification at *Step 5* , enter the Subject DN and click **Next**.

   Tip: If you have not yet defined the certificate in PingFederate or you do not know the DN, return to the previous screen and select Unanchored. Then click **Next** and click **Manage Certificates** on the SSL Verification Certificate screen to import the certificate, if needed, or to view its DN.
8. If you chose unanchored SSL certificate verification at *Step 5*, select the certificate to use for validating the SSL connection.

   If you have not yet imported the certificate into PingFederate, click **Manage Certificates**.
9. Click **Next**.
10. On the Summary screen, click **Done**.

Configure digital signature settings

This step defines the private key/certificate that you will use to sign assertions and SLO messages for this SP.

Note: Digital signing is required for SSO and SLO messages sent via POST or redirect bindings. Signing is not always required for profiles using the artifact or SOAP bindings.

The step applies to both IdP- and SP-initiated SSO and to either SLO profile (see *Choose SAML 2.0 profiles* on page 255) whenever outbound POST or redirect bindings are used. The step also is required for WS-Trust STS and for SSO if you chose to sign the SAML assertion, SAML response, or artifact resolution messages (see *Configure back-channel authentication* on page 290).

Note: This step does not appear if a connection configuration does not require it.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.

5. Click **Digital Signature Settings** on the Summary screen.

To specify a certificate:

1. Select the certificate from the drop-down list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing and decryption keys and certificates* on page 164).

   Note: For WS-Federation connections using JSON Web Tokens (see *Choose an SP connection type* on page 250), only EC and RSA certificates are supported. Furthermore, RSA certificates must have a minimum key size of 2,048 bits. The drop-down list filters out certificates not meeting these requirements for your convenience.

2. Optional: If you have agreed to send your public key with the SAML message, select the **Include this certificate's public key certificate ...** check box. To include the raw key in the signature as well, select the **Include the raw key …** check box.

   Note: For WS-Trust STS connections, by default the `<KeyInfo>` element in the SAML token includes a reference to the certificate rather than the full certificate unless this box is checked.

   Note: This step is not applicable to WS-Federation connections using JSON Web Tokens.

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

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 SP, SSO assertions, SAML 2.0 artifacts, or SOAP messages might also require signatures.

Whenever signatures are required, PingFederate provides a choice of trust models, including an option to use anchored signature-verification certificates embedded in incoming messages (see *Trust Models*). When this option is chosen in Signature Verification Settings, you must provide the Subject DN for embedded certificates coming from this partner, and the Issuer CA certificate must be part of the PingFederate trusted store (see *Manage trusted certificate authorities* on page 160).

Alternatively, you may choose to use unanchored certificates, in which case you must import your partner's public-key certificate during this configuration (or select it if it is already imported). To prevent any interruption of service due to an expired certificate, you can ask your partner for a new certificate in advance and import it as backup.

- To continue, click **Manage Signature Verification Settings**.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.

Choose a trust model

This screen allows you to choose the Trust Model you want to use for signature verification (see *Trust models* on page 74).

- Depending on the selection, the next step in this task varies:
  - For **Anchored**, the next step is to enter the Subject DN for your partner's certificate (see the next section, *Specify a Subject DN for anchored certificates* on page 293).

    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.
  - For **Unanchored**, the next step is to import your partner's certificate (see *Select unanchored certificates* on page 294).

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.
7. Click **Trust Model** on the Summary screen.

Specify a Subject DN for anchored certificates

When you choose to use an anchored certificate for signature verification, incoming SAML messages must contain the partner's verification certificate (see *Trust Models*). PingFederate verifies that the Issuer DN (if specified) matches that of one of the issuers in the chain, the Issuer CA is trusted and the embedded certificate's Subject DN matches the one specified on this screen. If so, PingFederate uses that certificate to verify the message signature.

To complete the configuration:

1. Enter the Subject DN or extract it from your SP partner's certificate if the certificate is stored on an accessible file system.

   Tip: To extract the Subject DN from a certificate, browse to select your SP partner's public certificate and click **Extract**.
2. Optional: Select the Restrict Issuer check box. Enter the Issuer DN or extract it from a certificate if it is stored on an accessible file system.

   Important: We recommend enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.

7. Click **Certificate Subject DN** on the Summary screen.

Select unanchored certificates

On the Signature Verification Certificate screen, you identify your partner's imported public certificate and, optionally, a secondary certificate to use when the first expires (see *Trust Models*).

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.
7. Click **Signature Verification Certificate** on the Summary screen.

To specify a verification certificate:

1. Select the certificate from the drop-down list.

   If you have not yet imported the certificate into PingFederate, click **Manage Certificates**.
2. Optional: Select a Secondary certificate for backup.

   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.

Review the signature verification settings

- Click any heading to change a setting, or click **Done** if the configuration is complete, then **Done** again on the Signature Verification Settings screen.

  Be sure to click **Save** on the Credentials screen if you are editing an existing connection.

Select an encryption certificate (SAML 2.0)

To enable XML encryption of all or part of an SSO assertion, you must identify the encryption certificate you will use (see *Specify XML encryption policy (for SAML 2.0)* on page 351).

You must also select a certificate if your requirements include encrypting an assertion in response to an attribute query (see *Specify security policy* on page 289).

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Select XML Encryption Certificate**.

   If this step is not present, you have chosen not to encrypt the assertion or the `SAML_SUBJECT` (see *Specify XML encryption policy (for SAML 2.0)* on page 351).

To identify the encryption certificate:

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 download web site* (`www.oracle.com/technetwork/java/javase/downloads/index.html`).

For more information about XML block encryption and key transport algorithms, see the "*XML Encryption Syntax and ProcessingW3C Recommendation*" (`www.w3.org/TR/xmlenc-core/`)

Note: As a 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, we recommend changing the selection for increased security.

2. From the drop-down list, select the applicable certificate and click **Next**.

   If the certificate is not in the list, click **Manage Certificates** to import it.

Select a decryption key (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 SLO requests from your partner may also be encrypted (if the connection uses SP-initiated SLO under SAML 2.0).

To enable XML encryption, you must identify a certificate for PingFederate to use to decrypt incoming SLO messages.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click a connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the SP Connection tab.
4. Click **Configure Credentials**.
5. Click **Select XML Decryption Key**.

   If this step is not present, you have chosen not to encrypt the assertion or the `SAML_SUBJECT` attribute (see *Specify XML encryption policy (for SAML 2.0)* on page 351).

To identify the decryption key:

1. Select an XML decryption key from the **Primary** list.
2. Optional: Select another XML decryption key from the **Secondary** list.

   If the desired XML decryption key is not in the list, click **Manage Certificates** to create or import it.
3. Click **Next**.

Review credential settings

From the Summary screen you can review or edit your credentials configuration.

Important: When you finish editing existing settings, you must click **Done** on the Summary screen and then **Save** on the Credentials screen. For a new connection, click **Done** and then click **Next** on the Credentials screen. Save the entire connection on the Activation screen (see *Review an SP connection settings* on page 304).

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 78).

Note: This configuration task is presented in the administrative console only when you enable Outbound Provisioning (see *Choose an SP connection type* on page 250).

- To continue, click **Configure Provisioning**.

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.

Note: The target configuration settings vary among SCIM outbound provisioning and various SaaS provisioning. The following steps describe the fields required for the bundled PingFederate provisioning plug-in for SCIM providers. For SCIM provisioning to *PingOne*, an administrator can find target information by setting up an Identity Repository and enabling provisioning, or by inspecting an existing setup. For SaaS Connector targets, please refer to documentation in your add-on distribution package, or *locate user guides online* (`documentation.pingidentity.com`).

1. **Note:** First obtain the Users Resource URL from your service provider—for example, https://example.com/v1/Users. If supported by the provider, also obtain the Groups Resource URL—for example, https://example.com/v1/Groups. Also determine whether the provider supports PATCH updates (see *Step 5* below) and/or prefers full distinguished name (DNs) for group provisioning (see *Step 6*).

   Enter the endpoint for managing users in the **Users Resource URL** field.
2. Optional: Enter the endpoint for managing groups in the **Group Resource URL** field.
3. Select the authentication method the endpoint accepts.

   Select **None** if the endpoint does not require authentication.
4. If you select either **Basic Authentication** or **OAuth 2.0 Bearer Token** as the authentication method, enter valid credentials in the **User** and **Password** fields for your provider.

   Additionally, if you choose **OAuth 2.0 Bearer Token** as the authentication method, in order to support the password grant type, enter valid credentials in the **Client ID** and **Client Secret** fields and the corresponding URL in the **Token Endpoint URL** field.
5. If the SCIM provider does *not* support PATCH updates, clear the associated check box.

   For information about PATCH, see the *SCIM specification* (`www.simplecloud.info/specs/draft-scim-api-01.html#edit-resource-with-patch`).
6. Optional: Depending on the target-provider needs, select the option to provision groups by supplying complete LDAP DNs, rather than only Common Names (CNs), to identify groups.

   Some SCIM providers, 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.
7. Select a deprovision method (`Delete User` or `Disable User`).

   Deprovisioning is triggered when previously provisioned users no longer meet the condition set in the **Source Location** screen.

   - Delete User removes the user account.
   - Disable User deactivates the user account.

   Note: For SaaS provisioning, the provisioner does *not* necessarily delete 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.
8. 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** in 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 Multi-Valued Attributes
- Complex Attributes
- Complex Multi-Valued 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 Multi-Valued	Not Applicable
`options`	Complex	`quantity`, `strike`, `first`, and `last`
`tablets`	Complex Multi-Valued	`model`, `serial`, and `type`. **Note:** `type` is a reserved sub-attribute for a complex multi-valued attribute.

Tip: For more information about SCIM and attribute types, see the web site *www.simplecloud.info*.

1. Specify the URI of the schema extension in the **Extension Namespace** field.

Tip: The default value is `urn:scim:schemas:extension:custom:1.0`. You can keep this value if your partner identifies custom attributes by this URI in its SCIM messages.

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**.
Set the attribute as a simple multi-valued attribute	1. Select the **Is Multi-Valued** check box. 2. Click **Done**.
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**.
Add sub-attributes and set the attribute as a complex multi-valued 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 Multi-Valued** check box. 3. Specify at least one value under the Types column for `type`, a reserved sub-attribute for a complex multi-valued 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**.

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.

In 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.

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 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 (see *Define an LDAP type*).

Information from your user-data store is used to supply mapped values for each user attribute required by the SP.

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 (see *Manage data stores*).
2. Click **Next**.

Modify source settings

The **Source Settings** screen shows the default configuration of the data store selected on the previous screen, including settings used by the PingFederate provisioner to determine when user information is added, changed, or removed. 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 multi-value 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 multi-value 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:

Field	Description
	Active Directory USN – For Active Directory only, this algorithm queries for update sequence numbers on user records that are larger than the last time records were checked. **Timestamp** – Queries for timestamps on user records that are *not older* than the last time records were checked. This check is more efficient from the point of view of the PingFederate provisioner but can be more time consuming on the LDAP side, particularly with the Oracle Directory Server. **Timestamp No Negation** – Queries for timestamps on user records that are *newer* than the last time records were checked. This algorithm is recommended for the Oracle Directory Server.
USN Attribute	The name of the attribute used to store the update sequence number—applicable when the Active Directory algorithm is chosen above.
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 Microsoft Active Directory or Oracle Directory Serve, 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 (see *Define an LDAP type*).

1. Optional: 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.

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 on the Source Location screen are grayed out. Support for the feature may become available in future Connector releases; please refer to documentation in your add-on distribution package, or *locate user guides online* (`documentation.pingidentity.com`) for more information.

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, described as follows:

1. Enter the base DN where user records are stored.

   PingFederate looks only at this node level or below it for user accounts (and groups when applicable) that need to be provisioned, based on the conditions set in the next step.
2. Specify group membership information or an LDAP filter to search for users (and groups when applicable) to be provisioned, as described in the following table:

Field	Description
Group DN	For Users, the group Distinguished Name associated with the user store, if applicable—required if a Filter is not used. For Groups (optional and subject to support from your partner or the SaaS Connectors), the DN of the higher-level group that contains the user groups, if applicable. **Important:** If a Group DN is used for Active Directory Groups, the domain functional level must be set as follows: • For Windows Server 2003 – Windows 2000 native or Windows Server 2003. • For Windows Server 2008 – Either of the above or Windows Server 2008. • For Windows Server 2012 – Any of the above or Windows Server 2012. Refer to Windows Server documentation or support for more information. (Optional) Select the Nested Search check box to include users (and groups) that are members of nested groups of the specified group. Nested group membership is preserved for SCIM provisioners and SaaS vendors if both the vendors and the SaaS Connectors support hierarchical structure in groups. **Note:** Nested Search is available when Active Directory or Oracle Directory Server is selected as the source user repository (see *Identify the source data store* on page 299).
Filter	An LDAP search filter. For Users, required if a Group DN is not provided. When configuring groups provisioning, an LDAP search filter is required if a group DN (for Groups) is not provided. If both Group DN and Filter are empty, no group is provisioned. For information about LDAP filters, refer to your LDAP documentation. Note that you may need to escape any special characters. **Important:** Group DN is ignored when an LDAP search filter is provided.

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.

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 data store 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 use of OGNL expressions may not be enabled for your PingFederate installation (see *Attribute mapping expressions*).

   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

In 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

1. Optional: Select the Root Object Class containing a user-store attribute 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 Default Value instead—skip this step and go to *step 5*. You can also do both for certain attributes, as needed: that is, specify LDAP attributes as well as a Default Value.
2. Under Attribute, select an attribute from the class.
3. Click **Add Attribute**.
4. 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.

5. Under Value Definition, enter or select a Default Value (optional, if one or more attributes is specified above).

   A drop-down 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.
6. 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.

   For information about the expression language supported by PingFederate, OGNL, see *Attribute Mapping Expressions*.
7. 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/None – Transforms the attribute value(s) to the case indicated, unless None 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.
8. Click **Done**.

Define mapping information for a custom attribute:

1. Select a sub attribute under Attribute ID.

   Note: Applicable only to Complex Attributes or Complex Multi-Valued Attributes (see *Specify custom SCIM attributes* on page 296).
2. Optional: Select the Root Object Class containing a user-store attribute 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 Default Value instead—skip this step and go to *step 5*. You can also do both for certain attributes, as needed: that is, specify LDAP attributes and a Default Value.
3. Select the source attribute under LDAP Attribute.
4. Optional: Select one or more processing Options, as defined below:

   Create Only – The field is provisioned only once and not subsequently updated.

   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/None – Transforms the attribute value(s) to the case indicated, unless None 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.
5. Optional: Enter a **Default Value**.

6. Click **Add Mapping**.

   Note: For complex attributes or complex multi-valued 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 140). 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 an SP 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 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 SSO or SLO transactions to or from this partner are disabled, as well as access to the WS-Trust STS for Web Service Clients associated with this connection.

Tip: The SSO Application Endpoint near the top of the Summary screen is an example URL that webmasters or web application developers at your site might use to invoke SSO for the connection. For details about SSO and other server endpoints, including optional query parameters, see *Application endpoints* on page 442.

To change a Connection Status:

- Select either Active or Inactive and then click **Save**.

To modify a connection setting:

1. If you know which step needs to be modified, click its link under the SP Connection tab.

   If you do not know where to change the setting, locate the currently configured data under one of the summary headings and then click the subheading above the data.

2. Change the information on the step screen and click **Save**, if available.

   If **Save** is not available, you are in the middle of a task (see *Tasks and steps* on page 27 in the *Get started with PingFederate* guide); click **Next** or **Done** until you reach a screen containing a **Save** button. Then click **Save** and continue as needed until you return to the Main Menu.

   If your modification requires related configuration changes, PingFederate provides error messages indicating the necessary steps and then guides you to the related screens (unless you click **Cancel**).

   Important: Be sure to click **Save** whenever that button appears, if you want to keep your changes.

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 67).

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 261).

You can create or modify an SP affiliation from the Main Menu under IdP Configuration or from a list of affiliations (see *Manage SP affiliations* on page 305).

To create an SP affiliation:

1. Click **IdP Configuration** on the Main Menu.
2. Click **Create New** under SP Affiliations.

Tip: See *Import affiliation metadata* on page 305 for subsequent steps.

To modify or delete an affiliation:

- See *Manage SP affiliations* on page 305.

Manage SP affiliations

You can manage SP affiliations on this screen.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
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 261).
- 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 136).

Once Auto-Connect is enabled on your PingFederate server, complete the configuration from the Main Menu under IdP Configuration. 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 assertion is considered valid. Each assertion has a time-stamp XML element and elements indicating the allowable lifetime of the assertion (in minutes) before and after the assertion time stamp.

Field Descriptions

Field	Description
Minutes Before	The amount of time before the assertion was issued during which it is to be considered valid.
Minutes After	The amount of time after the assertion 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 33 in the "Supported Standards" chapter of the *Get started with PingFederate* guide).

Note: The signing certificate is embedded in your server's Auto-Connect metadata (see *Using Auto-Connect* on page 75); 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 141). If you use a different certificate, ensure that it meets Auto-Connect validation requirements (see *Auto-Connect security model* on page 77).

To specify a certificate:

1. Select the certificate from the drop-down list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing and decryption keys and certificates* on page 164).
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 assertion creation* on page 257.

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 141).

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 until you return to the Main Menu.

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.

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. You must configure Server Settings from the Main Menu to establish your site as an SP before configuring connections to IdPs (see *Choose roles and protocols* on page 136).

Note that only one connection is needed per partner, even if you are integrating more than one web application. You can configure more than one connection, however, if your partner supports multiple protocols, or supports multiple federation IDs for the same protocol (see *Federation Server Identification*).

Note: This chapter 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 378.

Under some conditions, you can enable SSO for an unlimited number of partners at once by configuring a single, common connection (see *Using Auto-Connect* on page 75).

You can also deploy an IdP connection to bridge an identity provider to one or more service providers through one or more authentication policy contracts (see *Federation hub* on page 83 and *Deploying PingFederate as a federation hub* on page 86 for more information).

This chapter covers the following major topics:

- *SP application integration settings* on page 309
- *Federation settings* on page 316
- *Manage IdP connections* on page 318
- *Configure IdP Auto-Connect* on page 376

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 web site 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 64).

Under Application Integration Settings on the SP Configuration 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 Settings also include a link for configuring plug-in **Token Generators** (see *Manage token generators* on page 402).

Manage SP adapters

SP adapters are used to create a local-application session for a user in order to provide SSO access to your application(s) or other protected resources (see *SSO integration kits and adapters* on page 64). You can configure multiple instances of adapters (based on one or more adapters) to accommodate the varying needs of your IdP partners.

Note: If you are configuring an OpenToken Adapter, see *Configure the OpenToken IdP Adapter* on page 435.

SP adapter setup is available only if your server is configured as an SP (see *Choose roles and protocols* on page 136).

Important: If you install a new version of an adapter JAR file after setting up connections to instances of that adapter, you might need to reconfigure those connections. To find out, click each connection that uses the adapter (see *Access IdP connections* on page 318). Errors indicating reconfiguration points may be presented.

- You reach this screen by clicking **Adapters** under Application Integration Settings in SP Configuration.

To create a new adapter instance:

- Click **Create New Instance**.

  See the next section.

To edit an adapter instance:

- Click the Instance Name link.

To delete an adapter instance:

1. Click **Delete** next to the Instance Name on the Manage SP Adapter Instances screen. (To undo the deletion, click **Undelete**.)

   Note: This option is inactive if the instance is referenced elsewhere, or if it is a parent to other instances. To enable deletion, click **Check Usage** to locate the configurations that depend on the instance and go to each listed configuration to remove the dependencies. If the instance you want to delete is a parent, delete child instances first.
2. Click **Save** to confirm the deletion.

Create an SP adapter instance

On the Type screen, you begin creating an instance of an adapter that PingFederate uses for creating security sessions for your applications.

Field descriptions

Field	Description
Instance Name	A descriptive name for the adapter instance—for example, an identity management system name.
Instance ID	An internal identifier for the adapter instance. Must be alphanumeric with no spaces.
Type	A list of deployed adapter types available for creating an adapter instance for the server. A developer typically deploys any new adapter types before an administrator sets up a connection partner.
Parent Instance (Optional)	A list of configured instances for this type of adapter. If applicable, select an instance that can be used as a basis for a parent/child configuration.

To define an adapter instance:

1. Enter the Instance Name and Instance Id on the Type screen.
2. Select the Type from the drop-down menu.

   If the adapter you need is not listed, click **Visit PingIdentity.com for additional types** to see if a suitable adapter is available from the PingFederate download site. You can also create your own adapter (see *SSO integration kits and adapters* on page 64).
3. Optional: Select a **Parent Instance** from the list.

   If you are creating an instance that is similar to an existing instance, you might consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.
4. Click **Next** and enter information on subsequent screens for this adapter setup, as indicated in the following sections.

   Tip: The setup steps and information needed vary with the adapters deployed on your server (see *SSO integration kits and adapters* on page 64). For information about configuring the adapter packaged with PingFederate, see *Configure the OpenToken SP Adapter* on page 437.
5. Click **Done** on the Summary screen.
6. Click **Save** on the Manage SP Adapter Instances screen.

Configure an SP adapter instance

Configuration parameters on the SP Adapter Instance screen vary according to the adapter you choose. These options are controlled by the adapter plug-in software (see *SSO integration kits and adapters* on page 64).

- For information about configuring the OpenToken Adapter distributed with PingFederate, see *Configure the OpenToken SP Adapter* on page 437.
- For information about configuring an adapter contained in an integration kit, locate the user guide under *Product Documentation* at *pingidentity.com*.

Invoke SP adapter actions

Adapters can be written to perform configuration assistance or validation *actions*—for example, testing a connection to Active Directory. Actions may also include generation of parameters that might need to be set manually in a configuration file.

- For information about actions available using the OpenToken Adapter, see *OpenToken Adapter* on page 434.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click **Adapters** under Application Integration Settings.
3. Click an Instance Name.

4. Click **Actions** (if available).

To generate a properties list:

- Click **Download** under Action Invocation Link.

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.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click **Adapters** 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 SP Adapter Instances page.

Review an SP adapter configuration

From the Adapter Instance Summary screen, you can reach adapter settings for editing.

To edit the configuration:

1. Click the heading above the information you want to change.
2. Click **Save** on the configuration page and on the Manage SP Adapter Instances screen.

To save an adapter instance:

1. Click **Done** on the Summary screen.
2. Click **Save** on the Manage SP Adapter Instances screen.

Note: If this is the second adapter instance you have configured, then **Save** is not yet available; you must choose whether to map the new adapter instance to an application or resource URL. In this case, click **Next** to continue (see *Configure target URL mapping* on page 311 next).

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. 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.

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 needed for default situations, in cases where the target and adapter instance are not specified when the SSO/SLO is started (see *SP endpoints* on page 444). (When this information is provided with the SP request, the mapping table is ignored.).

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* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.)

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.

Field descriptions

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. If the 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 334), PingFederate tries the next entry.
Target Type	The type of the Target Session. If the IdP role is not activated (or is activated without any protocol for Browser SSO, such as SAML or WS-Federation), Target Type defaults to “SP Adapter”.
Target Session	A selection of configured SP adapter instances or SP connections. The available values depends on the chosen Target Type.

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://sp.company.com/acct101/`	`SP OpenToken`
`http://sp.company.com/*`	`Fedhub Cxn to SP`

A target URL of `http://sp.company.com/acct101/start` will be mapped to '`SP OpenToken`' 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 '`Fedhub Cxn to SP`' because the first mapping in the new configuration (`http://sp.company.com/*`) matches the target URL

Note that you can use only one wildcard (*) per URL.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click **Target URL Mapping** under Application Integration Settings.

To create target URL mappings:

1. Enter a URL
2. Select a Target Type. (Applicable only when the IdP role is activated with at least one protocol for Browser SSO, such as SAML or WS-Federation.).
3. Select a Target Session.
4. Click **Save**.

To edit target URL mappings:

1. Click **Edit** next to the Target Session. You can change the URL, Target Type and/or Target Session. (Target Type is only applicable when the IdP role is activated with at least one protocol for Browser SSO, such as SAML or WS-Federation.).

2. Click **Update**.
3. Click **Save**.

To delete target URL mappings:

1. Click **Delete** next to the Target Session.
2. Click **Save**.

   (Click **Cancel** to abort the deletion.)

To change the order of target URL mappings:

1. Click the up or down arrows at the left to rearrange the order.
2. Click **Save**.

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 539*).

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 360 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 Roles & Protocol screen (see *Choosing Roles and Protocols*).

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 **SP Configuration** on the Main Menu.
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 **SP Configuration** on the Main Menu.
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 **SP Configuration** on the Main Menu.
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 *Configuring Default Target URLs (Optional)*).

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 *Localization* on page 124. 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 endpoints* on page 444), but if none is provided, PingFederate will use the default values you enter on this screenunless, 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 120).

View SP application endpoints

Click **Application Endpoints** on the SP Configuration menu to see a list of endpoints and descriptions applicable to your federation role (IdP or SP). These endpoints are built into PingFederate and cannot be changed.

Web-application developers at your site need to know the application endpoints to initiate transactions via PingFederate (see *SSO integration kits and adapters* on page 64).

Note: For specific parameters required or allowed for Application Endpoints, see *SP endpoints* on page 444.

This screen also shows a Maintenance Endpoint that you can use to verify that the PingFederate server is running (see *System-services endpoints* on page 452).

Federation settings

If your identity federation uses the SAML 2.0 XASP profile (see *Attribute Query and XASP* in the "Supported Standards" chapter of the *Get started with PingFederate* guide), you may need to identify the IdP connection to which an attribute request applies. If so, click **Attribute Requester Mapping** under Federation Settings on the SP Configuration menu to complete the configuration (see *Manage attribute requester mappings* on page 316).

Also under Federation Settings, you can view protocol endpoints that your federation partners need to know to access your services via PingFederate.

Manage attribute requester mappings

If you are using the XASP profile, the application(s) at your site must supply the Subject Distinguished Name (DN) to identify a user's X.509 authentication certificate (see *Attribute Query and XASP* in the "Supported Standards" chapter of the *Get started with PingFederate* guide). 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: A `Format` query parameter must be set to a specified value for XASP (see */sp/startAttributeQuery.ping* on page 448).

On the Attribute Requester Mapping screen, you can map X.509 identifying information to connections and specify a default connection. You reach this screen from the Main Menu under Federation Settings.

Note: The Attribute Requester Mapping link does not appear on the SP Configuration menu unless you have enabled the SAML 2.0 protocol for the SP role (see *Choose roles and protocols* on page 136). You must also select the associated XASP check box.

At runtime, PingFederate tries to match the certificate's Issuer DN (if provided) against the list of Issuer DN(s), in the order shown on this screen, until a match is found. If no match is found, the server tries the Subject DN(s) in order. If no match is found, the Default connection is used.

For Issuer and Subject DNs, you can 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.

To map attribute requesters to connections:

1. Optional: Enter an Issuer DN when applicable, select a SAML 2.0 IdP Connection Name, and click **Add**.

   Repeat this step as needed for additional DNs.
2. Enter an Subject DN, select a SAML 2.0 IdP Connection Name, and click **Add**.

   Repeat this step as needed for additional DNs.
3. Select a Default IdP connection.

To edit a mapping:

1. Click **Edit** for the mapping in the Action column.
2. Make your changes and click **Update** in the Action column.
3. If you are editing an existing configuration, click **Save** to confirm the change.

To reorder the mapping list:

- Click the up or down arrow next to a DN.

To delete a mapping:

1. Click **Delete** for the mapping in the Action column.
2. If you are editing an existing configuration, click **Save** to confirm the deletion.

View SP protocol endpoints

Click Protocol Endpoints under Federation Settings in the SP Configuration section of the Main Menu to see a list of SAML, WS-Federation, and/or WS-Trust STS endpoints—a pop-up window displays only those endpoints related to the federation protocols enabled in Server Settings (see *Choose roles and protocols* on page 136). These endpoints are built into PingFederate and cannot be changed.

PingFederate provides a favorite icon for all Protocol Endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 128.

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 (see *Export metadata to an XML file* on page 101).

The table below describes each endpoint:

Table 13: PingFederate SP Endpoints

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**" footnote in this table.)
Metadata Service	`/` The default endpoint (empty path) from which partners can retrieve Auto-Connect metadata (see *Using Auto-Connect* on page 75).
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 421).
	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 402.

Service	URL and Description
	(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 *PingFederate properties* on page 112).

Note: For any connection using multiple virtual server IDs (see *Multiple virtual server IDs* on page 81), each virtual server ID has its own set of protocol endpoints. Use Metadata Export on the Server Configuration menu to export a connection metadata for your partner. For more information, see *Export metadata to an XML file* on page 101.

Manage IdP connections

As an SP, you manage connection settings to support the exchange of federation-protocol messages (SAML, WS-Federation, or WS-Trust) with an IdP or STS client application at your site.

Note: If you are configuring a new connection only for WS-Trust STS, follow the sections in this part of the manual up to and including *Identify the IdP* on page 324. Then turn to *WS-Trust STS configuration* on page 378.

These settings include:

- User attributes you expect to receive in an SSO assertion (including STS SAML tokens).
- User attributes that may be requested using the Attribute Query profile (if that profile is used).
- The protocol and, for SAML, the profile you will use, including detailed security specifications (the use of digital signatures, signature verification, XML encryption, and SSL). For more information, see *Supported standards* on page 28 in the *Get started with PingFederate* guide.

To continue with the configuration, you and your connection partner must have decided this information in advance (see *Federation planning checklist* on page 79). Your federation partner must supply some connection settings and other information (see *Configuration data exchange* on page 82).

Tip: If you are configuring connections to more than one partner under SAML 2.0 specifications, or if you intend to add partners in the future, consider using Auto-Connect (see *Configure IdP Auto-Connect* on page 376).

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 (see *Manage SP adapters* on page 309). In preparation for configuring a new SSO connection, you need to know which adapter instance to use (see *Manage target session mappings* on page 331). (No adapters are required for a connection that uses only the Attribute Query profile—see *Manage Attribute Query profile* on page 352.)

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 143).

Access IdP connections

You can create, modify or import connections directly under IdP Connections. Note that the SP Configuration menu displays the four most-recently modified connections. To view a list of all IdP connections, click the **Manage All** button.

From the Main Menu

From the Main Menu under SP Configuration, you can configure a new connection, modify an existing connection, import a connection, or view connections.

Tip: To copy or delete connections or to find connection drafts, click **Manage All** (see *From the Manage Connections screen* on page 319).

Note that long connection names are truncated for this display and the list is limited to four connections, chronologically ordered according to most recently edited. The full connection names and a complete list are displayed on the Manage Connections screen (see *From the Manage Connections screen* on page 319).

To begin configuring a new connection:

1. Click **SP Configuration** on the Main Menu.
2. Click **Create New** under IdP Connections.

> **Tip:** The fastest way to create a new connection is to click **Manage All** and copy the connection with similar settings (see *From the Manage Connections screen* on page 319).

To modify a connection:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Twenty-five most recently edited connections are displayed. To see all connections, including drafts, click **Manage All**.
3. On the Activation & Summary screen, click the heading for the information you want to change.
4. Make your change and click **Save**.

> **Note:** If **Save** is not available, it means your modification requires other changes or you are editing a screen that is part of a series of subtasks. Click **Next** and continue making indicated changes. The **Done** button indicates that further changes in the task are optional. When you have no further changes, click **Done** and then click **Save** on the task summary screen.

To import a connection:

1. Click **Import**.
2. In 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**.

From the Manage Connections screen

From the Manage Connections screen you can:

- Create a new connection.
- Modify or copy an existing connection.
- Continue working on a connection draft.
- Delete a connection—if it is not active or referenced in other parts of the configuration (In Use).
- Export or import individual connection configurations.

> **Note:** The connection export function results in an XML file that you can modify and import into the same PingFederate server or another PingFederate server acting in the same federation role (IdP or SP) at your site. You can import connections in this screen. Alternatively, you can use the Connection Management Service to complete the task (see *Importing connections* on page 467). Finally, you also have the option to automate this process (see *Automating configuration migration* on page 114).

- Export metadata about a connection to expedite your partner's corresponding configuration (see *Export metadata to an XML file* on page 101).
- Update a SAML connection by using a metadata file or a metadata URL from your partner.

> **Important:** The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.

On this screen you can also globally override transaction logging levels set for individual connections or restore connection-based logging (see *Runtime transaction logging* on page 93).

Tip: A check-mark icon next to a Connection Name indicates that the connection has been checked for configuration errors. For more information about connection-validation features associated with this screen, see *Manage IdP connection validation* on page 321.

To access the Manage Connections screen:

1. Click **SP Configuration** on the Main Menu.
2. Click **Manage All** under IdP Connections.

To begin configuring a new IdP connection:

- Click **Create Connection** on the Manage Connections screen.

To copy a connection:

1. Click **Copy** under Action for the connection you want to copy.
2. Update the Connection ID (of the partner) and the Connection Name in the General Info screen (see *Identify the IdP* on page 324).
3. Make any further changes needed for the new connection.

To edit a connection or continue working on a draft:

- Click the Connection Name link.

  For a draft, you will return to where you left off.

To export a connection:

1. Click **Export Connection** under Action for the connection.
2. Save the XML file on your file system.

   You can change the name of the file, but keep the XML extension.

To import a connection:

1. Click **Import**.
2. In 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**.

To export connection metadata:

1. Click **Export Metadata** under Action for the connection.

   This action takes you to the Export Metadata screen flow, with the connection selection preset (see *Export metadata to an XML file* on page 101).
2. Complete the steps remaining in the Export Metadata screen flow (starting at *Step 4* under *Provide SAML metadata by file* on page 101).

To update a SAML connection with your partner's metadata:

1. Click **Update with Metadata** under Action for the applicable SAML connection.
2. Follow the workflow to complete the update process.

Important: The update operation may require additional configuration. Reviewing the connection after the update operation is recommended.

To delete a connection:

1. Under Action, click **Delete** for the connection.

   (To undo the deletion, click **Undelete**.)

   Note: The **Delete** function is not available if the connection is Active or In Use.

2. To confirm the deletion, click **Save**.

To sort the list of connections:

- Click the arrow next to any column heading to sort the list based on that column.

  Note: The Virtual ID displays the default virtual server ID of the connection, if any (see *Identify the IdP* on page 324).

To filter the list by Protocol and/or Status:

- Select a filter criterion from either or both of the drop-down lists.

To override connection-based transaction logging:

1. Select **On** under Logging Mode Override.
2. Choose the logging mode you want to use for all connections.

To restore connection-based transaction logging:

- Select **Off** under Logging Mode Override.

Manage IdP connection validation

By default PingFederate automatically validates all existing connections before displaying the Manage Connections screen. This validation ensures that any updates to supporting components—adapters or data-store configurations, for example—have not invalidated any connection settings.

If such errors are found, a warning icon appears next to the Connection Name.

To correct errors:

- Click the Connection Name to reach the top-level task in which reconfiguration is needed, and to see the error message. Then navigate into deeper tasks using **Configure . . .** buttons to find a link to the screen that needs updating.

  (For more information about console navigation, see *About Tasks and Steps* in the "Console Navigation" chapter of the *Get started with PingFederate* guide).

Note that the connection validation time increases with the number of connections and when connections are configured to access data stores for adapter mapping. Consequently, there may be noticeable delays in displaying the Manage Connections screen. For this reason, PingFederate provides a way to turn off the automatic validation under Server Settings (see *Configure system options* on page 138).

When validation is turned off, administrators can check connections manually on the Manage Connections screen. A question-mark icon indicates the connection has not been validated. You may, however, still edit the connection by clicking its name.

In addition, Action links are disabled (except for **Delete**, if the connection is Inactive and/or In Use) until the connection is validated.

When automatic validation is disabled, use one of the following procedures to validate connections:

To validate a *single* connection:

- Click icon next to the Connection Name.

Note: Validating a single connection does not check connectivity for any configured data-store lookups (see *Choose an attribute mapping method* on page 334). However, this check *is* performed when you access the connection for editing.

To validate *all* connections (including for data-store connectivity):

1. Click **Check All Connections for Errors**.
2. When prompted, click **Yes**.

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, and/or Inbound Provisioning (see *Connection types* on page 57).

Note: You can add STS, OAuth, and Inbound Provisioning support to any existing SSO connection, or vice versa, at any time.

If your federation deployment supports multiple protocols , then for new SSO connections you can also select the applicable protocol on the Connection Type screen (see *Choose roles and protocols* on page 136).

Note: If your partner's deployment also supports multiple protocols and you intend to communicate using more than one, then you must set up a separate connection for each protocol.

- To configure a connection for secure browser-based SSO, select Browser SSO Profiles and the Protocol (if necessary).
- To configure a connection for WS-Trust STS, make that selection.
- To configure an OAuth SAML Grant connection, make that selection.

  Note: This option is available only after you configure an access token plug-in (see *OAuth access token management* on page 186).
- 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: The Inbound Provisioning option is active only after you enable the Inbound Provisioning protocol on the Roles & Protocol screen (see *Choose roles and protocols* on page 136).
- 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, you can choose to enable JIT Provisioning in conjunction with Browser SSO (see *Just-in-time provisioning* on page 79).

You can also choose to map user attributes for persistent grants used by the optional PingFederate OAuth Authorization Server (see *PingFederate OAuth AS* on page 60).

Note: The OAuth Attribute Mapping option is active only when the OAuth 2.0 Authorization Server (AS) role is enabled on the Roles & Protocol screen (see *Choose roles and protocols* on page 136).

For SAML 2.0, you also have the option of configuring the Attribute Query profile, with or without SSO (see *Attribute Query and XASP* on page 42 in the “Supported Standards” chapter of the *Get started with PingFederate* guide).

Note: This screen is presented only for browser-based SSO connections (see *Choose an IdP connection type* on page 322).

- To create a connection for browser-based SSO, ensure that Browser SSO is selected and click **Next**.

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 in the **Server Configuration** > **Server Settings** > **Metadata Lifetime** screen (see *Configure metadata lifetime* on page 141).

Although optional, it is recommended that you turn on email notifications for SAML metadata update events in the **Server Configuration** > **Server Settings** > **Runtime Notification** screen (see *Configure runtime notifications* on page 130).

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. In 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 in 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** workflow, the administrative console reminds you to select it from the **Existing URL Name** list. 3. (Optional) Clear the **Enable Automatic Reloading** check box to disable automatic update.

Metadata medium	Steps
	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 in the **Import Certificate** screen, and then click **Next**. **5.** In 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.** (Optional) 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** workflow (see *Manage metadata URLs* on page 171). **5.** Click **Next**.

Identify the IdP

On the General Info screen, you provide a required unique identifier and display name for a connection, as well as optional contact information. In addition, on this screen you can define a default error message that end users will see in the event that SSO fails, and you can set the level of transaction logging for this connection partner (see *Runtime transaction logging* on page 93).

Field Descriptions

Field	Description
Partner's Entity ID/ Issuer/ Partner's Realm (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 Issuer your partner advertises. For a WS-Federation connection, this is your partner's Realm. This ID may have been obtained out-of-band or via a metadata file if you are using a SAML protocol (see *Export metadata to an XML file* on page 101). For STS-only connections, 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 under Server Settings (see *Specify federation information* on page 137), enter a virtual server ID in this field and click **Add**. Enter additional virtual server IDs as needed (see *Multiple virtual server IDs* on page 81).

Field	Description
Base URL	The fully qualified hostname and port on which your partner's federation deployment runs (e.g., `https://sso.theidpcompany.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	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 (see *Customizable user-facing screens* on page 120). 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 *Localization* on page 124. 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 (see *Runtime transaction logging* on page 93).

For a new connection:

- Fill in the information needed and click **Next**.

  Connection ID and Connection Name are required (see "Field Descriptions" above).

  Note that the values in Virtual Server IDs identify your own federation deployment for this connection only and override the ID you specified under Server Settings (see *Federation Server Identification*).

For an existing connection:

- If you are editing existing information, modify the fields as needed and click **Save**.

Configure Browser SSO

Browser-based SSO (also, Browser SSO) is another term for secure SSO, which relies on a user's web browser and HTTP to broker XML identity-federation messaging 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**.

SP 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 *Supported standards* on page 28 chapter of the *Get started with PingFederate* guide). 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 PingTrust 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.

Note: The configuration screens represented in this chapter show "SAML 2.0" in their left corners, unless they are exclusive to WS-Federation or SAML 1.x setup requirements. When the SAML 2.0 screens are also applicable to SAML 1.x and/or WS-Federation connections, the SAML 2.0 representations and discussions also apply to the other protocols, unless otherwise indicated.

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 369). Also, other configuration tasks may remain to be configured for new or modified connections, depending on selected connection options (see *Choose IdP connection options* on page 322).

Important: For new connections you must completely configure these SSO settings and subsequent tasks before you can save the connection on the Activation & Summary screen. Until then, the *configuration is temporary and can be lost*; the console times out after several minutes of inactivity. At any time, however, you can click **Save Draft**, which is available on most screens after you enter General Information (see *Console buttons* on page 27 in the "Console Navigation" chapter of the *Get started with PingFederate* guide).

Use the lists and links (or page references) below to find specific information about steps that may apply to your SSO connection requirements:

SAML 2.0 SSO steps

- *Choosing SAML Profiles*
- *User-Session Creation*
  - *Choose an identity mapping method* on page 329
  - *Define an attribute contract* on page 330
  - *Manage target session mappings* on page 331
- *Manage protocol settings* on page 346
  - *Specify SSO service URLs (SAML)* on page 346
  - *Define SLO service URLs (SAML 2.0)* on page 348
  - *Select allowable SAML bindings (SAML)* on page 349
  - *Specify an artifact lifetime (SAML)* on page 349
  - *Define artifact resolver locations (SAML 2.0)* on page 350
  - *Configure signature policy* on page 351
  - *Specify XML encryption policy (for SAML 2.0)* on page 351

WS-Federation SSO steps

- *Configure user-session creation* on page 328
  - *Choose an identity mapping method* on page 329
  - *Define an attribute contract* on page 330
  - *Manage target session mappings* on page 331
- *Manage protocol settings* on page 346
  - *Specify a service URL (WS-Federation)* on page 347

SAML 1.x SSO steps

- *Select SAML 2.0 profiles* on page 327
- *Configure user-session creation* on page 328
  - *Choose an identity mapping method* on page 329
  - *Define an attribute contract* on page 330
  - *Manage target session mappings* on page 331
- *Manage protocol settings* on page 346
  - *Specify SSO service URLs (SAML)* on page 346
  - *Select allowable SAML bindings (SAML)* on page 349
  - *Define artifact resolver locations (SAML 2.0)* on page 350
  - *Configure signature policy* on page 351

Select SAML 2.0 profiles

A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use (see *Federation planning checklist* on page 79). For SAML 2.0, PingFederate supports all IdP- and SP-initiated SSO and SLO profiles.

The SAML 1.x implementation supports standard IdP-initiated SSO and nonstandard SP-initiated SSO.

For information on typical SSO/SLO profile configurations, including illustrations, see *Browser-based SSO* on page 30in the *Get started with PingFederate* guide..

You can configure profiles individually or all together. PingFederate presents the correct configuration steps to fit your choices. Steps that apply to one SSO or SLO profile often apply to others and are reused automatically across profiles.

Note: 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 (see *SAML 1.x Profiles*in the *Get started with PingFederate* guide.).

For WS-Federation, the SAML Profiles screen is not presented.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **SAML Profiles** on the Summary screen.

To configure profiles:

1. Select the profile(s) applicable to this connection and click **Next**.

   For SAML 2.0 connections, you must select an SSO profile before you can enable SLO.
2. Continue through the remaining connection-configuration tasks.

The following topics provide more information on requirements for each SAML profile:

- *About IdP-Initiated SSO* on page 327
- *About SP-Initiated SSO* on page 327
- *About IdP-Initiated SLO* on page 328
- *About SP-Initiated SLO* on page 328

About IdP-Initiated SSO

When PingFederate is operating as an SP, the IdP-initiated SSO profile configuration defines the message-transport mechanisms (bindings) your enterprise has agreed to allow for receiving SAML assertions, plus any digital signature verification requirements for inbound assertions (see *Security Infrastructure*).

For illustrations of typical profile and binding scenarios, see *Browser-based SSO* on page 30in the *Get started with PingFederate* guide..

For this configuration you need to know:

- The transport binding(s) to which you and your partner have agreed
- The certificate to be used for verifying incoming digital signatures from your IdP (optional for the artifact binding)

When Artifact is an allowable inbound SAML binding, you also need to know the endpoint(s) to your partner's Artifact Resolution Service(s) and the SOAP client authentication mechanism to use: either HTTP Basic, SSL client certificates, a digital signature, or a combination of these mechanisms.

About SP-Initiated SSO

The SP-initiated SSO profile configuration defines the message-transport mechanisms (bindings) and security requirements for sending authentication requests and receiving assertions when your site initiates SSO transactions.

For SAML 1.x, the SP-initiated SSO profile is also known as the “destination-first” profile, which was added as a supported “non-normative” use case after the release of the SAML 2.0 specifications.

For illustrations of typical profile and binding scenarios, see *Browser-based SSO* on page 30in the *Get started with PingFederate* guide..

For this configuration you will need to know:

- The endpoint URL(s) for your IdP's Single Sign-on Service(s)
- The transport bindings to which you and your partner have agreed (inbound and outbound)
- The certificates you will use to sign outbound authentication requests and to verify incoming digital signatures from your IdP

When Artifact is an allowable inbound SAML binding, you also need to know the endpoint(s) to your partner's Artifact Resolution Service(s) and the SOAP client authentication mechanism to use: either HTTP Basic, SSL client certificates, a digital signature, or a combination of these mechanisms.

About IdP-Initiated SLO

The SAML 2.0 IdP-initiated SLO profile configuration defines the message-transport mechanisms (bindings) and security requirements that you and your partner have agreed upon for exchanging SLO requests and responses.

Note: SLO is not supported by the SAML 1.x specifications.

For more information about SLO, see *Single logout* on page 41 in the “Supported Standards” chapter of the *Get started with PingFederate* guide.

For this configuration you need to know:

- The transport bindings that you and your partner have agreed upon to send SLO requests and receive responses
- The certificates to be used for signing outgoing messages and for verifying incoming digital signatures from your IdP (optional for the artifact binding)
- The URL(s) of your IdP's Single Logout Service(s)
- The URL of your IdP's Artifact Resolution Service(s) (to resolve artifacts from the IdP) and SOAP client authentication requirements

About SP-Initiated SLO

The SAML 2.0 SP-initiated profile configuration for SLO defines the message-transport mechanisms (binding)s and security requirements that you and your partner have agreed upon for exchanging SAML requests and responses.

Note: SLO is not supported by the SAML 1.x specifications.

For more information about SLO, see *Single logout* on page 41 in the “Supported Standards” chapter of the *Get started with PingFederate* guide.

For this configuration you need to know:

- The transport bindings that you and your partner have agreed upon to send SLO requests and receive responses
- The certificates to be used for signing outgoing messages and for verifying incoming digital signatures from your IdP (optional for the artifact binding)
- The URL(s) of your IdP's Single Logout Service(s)
- The URL of your IdP's Artifact Resolution Service(s) (to resolve artifacts from the IdP) and SOAP client authentication requirements

Configure user-session creation

As an SP, you must specify how you use information sent from the IdP in SSO assertions 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 must associate one or more authentication policy contracts to the IdP connection (see *Federation hub* on page 83 and *Deploying PingFederate as a federation hub* on page 86 for more information).

For both scenarios, the configuration includes:

- Choosing an identity-mapping method (see *Choose an identity mapping method* on page 329).
- Defining the attribute contract you will use with this partner, if any (see *Define an attribute contract* on page 330).
- Configuring instances of one or more target sessions (see *Manage target session mappings* on page 331) and specifying how they are used to fulfill the contract.
- 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 66). At the Identity Mapping step, you choose which method to use with a particular IdP connection. You and your partner may want to decide in advance which option to use (see *Federation planning checklist* on page 79).

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 *About attributes* on page 67).

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—then select **Account Mapping**.

  Account mapping uses the value passed in the SAML assertion's `SAML_SUBJECT` 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 353).
- If you want to create a long-term association between a remote user and a local account, then select **Account Linking** on the Identity Mapping screen.

  To set up an attribute contract to use in conjunction with account linking, select the check box next to "The assertion includes attributes . . ." after selecting **Account Linking**.

  Tip: An SP PingFederate uses a default, HyperSQL database to handle account linking. You can use your own database instead, as needed. For more information, see *Configure an LDAP connection* on page 145.
- If you have selected only the SP-initiated SSO profile (with or without the IdP-initiated SLO profile, the SP-initiated SLO profile, or both) 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.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Identity Mapping** on the Summary screen.

Define an attribute contract

An attribute contract is the set of user attributes that you and your partner have agreed will be sent in SAML assertions for this connection (see *Attribute contracts* on page 68). You identify these attributes on this screen.

`SAML_SUBJECT` is always sent in a SAML assertion and contains the name identifier of the user requesting SSO. When you select account mapping as the identity mapping technique, the `SAML_SUBJECT` is available to help map the incoming user to a local ID on your system (see *Choose an identity mapping method* on page 329).

For account linking, the `SAML_SUBJECT` contains an identifier that the SP server uses to make a permanent association between the remote user and a local account. The `SAML_SUBJECT` itself is not available to the SP application and thus does not appear in the Attribute Contract on this screen.

Optionally, you can mask the values of attributes (other than `SAML_SUBJECT`) in the log files that PingFederate writes when it receives assertions (see *Attribute masking* on page 71).

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.

   Tip: If you are configuring a SAML connection to an InCommon participant (see *www.incommonfederation.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**.

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.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Attribute Contract** on the Summary screen.

   If this step is not in the list, then you have chosen to use account linking and specified that the IdP is not including additional attributes in the assertion (see *Choose an identity mapping method* on page 329).

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.

The PingFederate server for an SP uses integration adapters to identify the user to your applications based on attributes sent in an assertion. The server uses this information to create a local session that enables access to user-requested resources (the “target”) at your site. (See *SSO integration kits and adapters* on page 64.)

Each adapter instance requires a set of attributes into which you map values found in the assertion. You can map additional attributes, as needed, from local data stores, or you can use static or variable text values. An adapter instance will fail to create a local session for the incoming user if it is unable to find values for each of its required attributes.

You may also deploy an IdP connection to bridge an identity provider to one or more service providers.

Tip: In this scenario, PingFederate is a federation hub for both parties. (See *Federation hub* on page 83 and *Bridging multiple IdPs to an SP* on page 84 for more information.)

PingFederate uses authentication policy contracts to associate this IdP connection with the applicable SP connections to the service providers (see *Deploying PingFederate as a federation hub* on page 86).

Each authentication policy contract has its own set of attributes which you map values from the assertions.

You must associate at least one target session, 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—then you should map multiple adapter instances. If you are bridging an identity provider to multiple service providers, you should map multiple authentication policy contracts.

When target sessions are restricted to certain virtual server IDs, the allowed IDs are displayed under the Virtual Server IDs column.

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 311).

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** on the Summary screen.

To begin mapping an adapter:

- Click **Map New Adapter Instance** and follow the configuration screens (see the following sections for more information).

To begin mapping an authentication policy contract:

- Click **Map New Authentication Policy** and follow the configuration screens (see the following sections for more information).

To begin modifying an existing target session mapping:

- Click the **Adapter Instance Name** or the **Authentication Policy Contract Name**, and navigate through the steps to the information you want to change.

Select a target session

Use this screen to associate an SP adapter instance or an authentication policy contract with an IdP connection.

Tip: An SP adapter instance is available for use within an IdP connection only after it has been deployed and configured in PingFederate.

To select an adapter instance:

- Choose an adapter instance from the drop-down menu and click **Next** to continue.

  (Optional) To override any SP adapter instance, select the **Override Instance Settings** check box (see *Override an SP adapter instance* on page 332).

  If the adapter instance you need is not available, click **Manage Adapter Instances** to define one or more adapter instances you need for this connection.

  Note that an adapter instance can be mapped only once per connection. However, the same adapter instance may be mapped by multiple connections.

  Tip: Adapter contracts for some adapters can be customized for individual connection requirements (see *Manage SP adapters* on page 309).

To select an authentication policy contract:

- Choose an authentication policy contract from the drop-down menu and click **Next** to continue.

  If the authentication policy contract you need is not available, click **Manage Authentication Policy Contracts** to define one or more authentication policy contracts you need for this connection.

  Note that a particular authentication policy contract can be mapped only once per connection. You may however map the same contract to multiple connections.

Override an SP adapter instance

Overrides at the connection level simplify adapter management by allowing you to create a small set of adapter instances that define the base configuration requirements and then overriding settings at the connection level as needed.

You may override any SP adapter settings at the connection level either during or after connection mapping.

Alternatively, you can override any adapter instance to apply to several connections, see *Hierarchical plug-in configurations* on page 66 for more information about adapter overrides.

Note: Any changes to the base 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, make your changes, and then click **Next**. When you are finished, click **Done** to continue with SP adapter mapping.

  Note: Override adapter-setting screens are functionally identical to those used for creating a new adapter connection. Refer to the table below to find sections in this manual containing configuration information and procedures.

  The display of some screens listed in the table depends on the type of adapter you are configuring.

  For information about the override adapter-setting screens, depending on the type of setting, use the following table:

Override Screen	Manual Section
Adapter	See *Configure an SP adapter instance* on page 310.
Actions	See *Invoke SP adapter actions* on page 310.
Extended Contract	See *Extend an SP adapter contract* on page 311.

- To remove any adapter connection override, clear the **Override Instance Settings** check box, and then complete the rest of the setup.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Adapter Instance** on the Summary screen.
8. Select the **Override Instance Settings** check box.
9. Click **Next**.

View the SP adapter type

Adapter override instance type information.

This screen is view only. This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Manage SP adapters* on page 309.

Override SP adapter settings

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Configure an SP adapter instance* on page 310.

Override the SP adapter actions

- Make any changes, and then click Next.

This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Invoke SP adapter actions* on page 310.

Override the SP adapter extended contract

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

This screen is functionally identical to the one used for creating a new adapter instance. For more information about this screen, see *Extend an SP adapter contract* on page 311.

Review the overridden SP adapter instance

- On the Override Instance Settings Summary screen, click any heading to change your processor configuration; or click **Done** to continue.

Restrict a target session to certain virtual server IDs

When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), you have the option to enforce authentication requirements by restricting a target session (an adapter or an authentication policy contract) to certain virtual server IDs. This optional setting can be applied to each target session added to the connection using virtual server IDs. By default, no restriction is imposed.

To restrict a target session to a subset of the available virtual server IDs:

1. Select the **Restrict Virtual Server IDs** check box.
2. In the **Allowed Virtual Server IDs** area, select virtual server IDs that you want to allow for this target session.
3. Click **Next**.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** on the Summary screen.
8. Click the target session name.
9. Click **Virtual Server IDs** on the Summary screen.

Note: The Virtual Server IDs screen is only available for connections using at least one virtual server ID, see *Federation Server Identification*.

Choose an attribute mapping method

To populate the attributes required by the target session (an adapter contract or an authentication policy contract), you can use values supplied by SAML assertions from the IdP exclusively, or in addition to values retrieved from local user-data stores (see *Manage data stores* on page 143).

- If you choose to look up additional values, click the applicable button and then **Next**. This selection allows you to identify data sources and specify lookup queries in subsequent screens (see *Configure an attribute source* on page 335 next).

  Or:

If you choose not to look up additional values, click the applicable button (if it is not already selected) and then **Next**. This selection takes you directly to a screen where you can map attribute values from the assertion (see *Configure target session fulfillment* on page 340).

Tip: To determine whether you need to look up additional values, compare your attribute contract against the contract of your target session (see *Define an attribute contract* on page 330 and *Select a target session* on page 332). If target session requires more information, determine whether your local data stores can supply it. (You can also choose to use text constants or expressions for certain information—see *Configure target session fulfillment* on page 340.)

Configure an attribute source

For data-store setup information, refer to the sections indicated in the following steps.

Note: As you make selections on configuration screens, ensure that you allow enough time for PingFederate to access your data store and populate drop-down lists.

1. See *Choose a data store* on page 335.
2. See the following sections in this manual, depending on the type of data store:

Data Store Type	Related TopicManual Section
JDBC	• *Choose a database table and columns* on page 336 • *Define a database filter* on page 337
LDAP	• *Configure an LDAP directory search* on page 338 • *Define an LDAP filter* on page 339
Custom	• *Configure custom data-source filters* on page 340 • *Select custom data-source fields* on page 340

3. See *Configure target session fulfillment* on page 340.

Choose a data store

This screen allows you to choose a data store from a previously configured list (see *Manage data stores* on page 143). Attribute values extracted from this data store are used in combination with the values from the attribute contract to fulfill the adapter contract for this adapter instance or the authentication policy contract (see *Manage target session mappings* on page 331).

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **Data Store** on the Summary screen.

   If this step is not present, then the use of a data store has not been selected (see *Choose a database table and columns* on page 336).

To define an attribute source:

- Choose an Active Data Store and click **Next**.

  A data-store configuration must be defined under System Settings for use within a connection. If the data store you want is not shown in the drop-down menu, click **Manage Data Stores** to add it (see *Manage data stores* on page 143).

Choose a database table and columns

When you choose to use a database source for attributes, you follow this path through the configuration steps.

On this screen you begin to specify exactly where additional data can be found to complete the attribute contract when you send an assertion to this IdP (see *Define a database filter* on page 337). Only one table may be used as a source of data for a JDBC lookup.

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 *http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html*.

Field descriptions

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 a JDBC query. Other databases, such as MySQL, do not require selection of a schema.
Table	Displays the table(s) contained in the database. Select the table to retrieve data from the data store.
Columns to return from SELECT	Displays the columns available from the selected table. Select the columns that are associated with the desired attributes you would like to return from the JDBC query.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **Database Table and Columns** on the Summary screen.

   If this step is not shown, this connection is not yet configured to use a database to look up attributes. For information about changing this configuration, see *Choose a data store* on page 335.

To select a database table and columns for queries:

1. Choose a Schema file (when applicable) from the drop-down list.
2. Choose a Table from the drop-down list.
3. Choose a name under Columns to Return from Select and click **Add Attribute**.

Tip: Click **Refresh** if you are updating an existing configuration and changes may have been made to the database.

Repeat this step for other columns as needed.

Note: You do not need to add a column here for it to be used as part of a search filter (see *Define a database filter* on page 337 next). Add only attributes from which you need actual values to pass to the target session.

Define a database filter

On this screen you begin to specify exactly where additional data can be found to complete the attribute contract when you receive an assertion from this IdP (see *Creating an Attribute Contract*).

The JDBC `WHERE` clause queries your data store to locate a user record. Once the record is located, the configured SELECT statements retrieve the attribute values.

The clause is in the form:

```
WHERE column1=value1 [AND column2=value2] [OR ...]
```

The left side of the first variable pair uses a column name in the database table you selected (see *Choose a database table and columns* on page 336).

The right side generally uses values passed in from the assertion. Possible variables for these, including the correct syntax, are listed under Assertion Values.

You can also apply additional search criteria from your own database, using any other columns from the targeted table.

Tip: Click "**View List of Columns . . .**" to see a list from which to copy and paste.

Example:

```
userid='${username}'
```

In this example `userid` is the name of a column in the JDBC data store. On the right side, `'${username}'` returns the value of the `username` from the assertion.

Important: You *must* use the `${}` syntax to retrieve the value of the enclosed variable and use single quotation marks around the `${}` characters.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **Database Filter** on the Summary screen.

   If this step is not shown, this connection is not yet configured to use a database to look up attributes. For information about changing this configuration, see *Choose a data store* on page 335.

To construct the `WHERE` clause:

1. Enter the statement in the space provided, following the guidelines and example above.

The initial `WHERE` is optional.

2. Ensure the syntax and variable names are correct.

   When you click **Next**, you will map attribute values returned from the database into the attribute contract (see *Choose an attribute mapping method* on page 334).

Configure an LDAP directory search

When you choose to use an LDAP source for attributes, you follow this path through the configuration steps.

On this screen you begin to specify exactly where additional data can be found to supply to the target session (an SP adapter or an authentication policy contract) in order to access a resource on your system.

Field descriptions

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	Determines the node depth of the query. Select Subtree, One level or Object.
Root Object Class	Specifies the object type within the LDAP hierarchy from which attributes will be returned.
Attribute	The available attribute names for the selected directory structure. Select the names associated with the attributes that you would like to return from the query.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **LDAP Directory Search** on the Summary screen.

   If this step is not shown, this connection is not yet configured to use LDAP to look up attributes (see *Choose a data store* on page 335).

To select LDAP attributes:

1. Optional: Enter a Base DN.
2. Select a Search Scope.
3. Select a Root Object Class.
4. Under Attributes to return from search, choose an attribute and click **Add Attribute**.

   Note that the attribute Subject DN is always returned by default.

   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 as well as indirectly through nested group membership (if any) under the Base DN.

For example, suppose you have three groups under the Base DN, namely 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).

5. Repeat the last step for other attributes as needed.

Note: You do not need to add an attribute here for it to be used in a search filter (see *Define an LDAP filter* on page 339). Add only attributes from which you need values to map to the target session.

Define an LDAP filter

The LDAP filter queries the data store to retrieve a user record associated with a particular value (or values) from the assertion. The filter is in the form:

```
(attribute=${value})
```

The left-side variable is an attribute from the data store (see *Configure an LDAP directory search* on page 338).

The right side generally uses values passed in from the assertion.

You can also apply additional search criteria from your data store, using any other attributes from the targeted object classes.

Tip: Click "**View List of Available LDAP Attributes**" for a list from which you can copy and paste.

For general information about search filters, consult your LDAP documentation.

Example:

```
(UNAME=${username}).
```

In this example `UNAME` is the name of an attribute in the LDAP data store. On the right side, `${username}` returns the value of `username`. in the assertion.

Important: You *must* use the `${}` syntax to retrieve the value of the enclosed variable.

Field description

Field	Description
Filter	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 at least three components: 1) at least one attribute (attribute data type) to search on, 2) a search filter operator that will determine what to match, and 3) the value of the attribute being sought. Searches must have at least one of each of these three components.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.

6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **LDAP Filter** on the Summary screen.

   If this step is not shown, this connection is not configured to use LDAP to look up attributes (see *Choose a data store* on page 335).

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.

   Note: If you used an anonymous binding to create this LDAP connection, your access might be restricted (see *Configure an LDAP connection* on page 145).
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

Configure custom data-source filters

When you choose to use a custom source for attributes, you follow this path through the configuration steps.

On this screen you specify a filter, or lookup 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.

Select custom data-source fields

On the Configure Custom Source Fields screen, you can choose from among the fields shown to map to the target session (an adapter contract or an authentication policy contract). These choices are supplied by the driver implementation. Select only those needed to fulfill the target session.

Configure target session fulfillment

The last step in configuring a target session is to map each attribute required to a value (see *Choose an attribute mapping method* on page 334). If you integrate your applications with PingFederate through an SP adapter, these are the values that will be used to create a local session; an SSO operation fails if the SP is unable to fulfill the mapping requirements defined here. If you are bridging an identity provider to a service provider, these are the values that will be used to fulfill the authentication policy contract, which in turn will be used by the SP connection to create the assertions for the service provider (see *Federation hub* on page 83 for more information).

Map attributes from one of these Sources:

- Account Link

  This source appears only if you have elected to use account linking for an SP adapter (see *Choose an identity mapping method* on page 329). When you make this selection, the associated Value drop-down list is populated with Local User ID. Normally, you would map this identifier to target an adapter attribute that represents the local user ID. This source is not applicable to authentication policy contracts.
- Assertion

  Values are contained in the assertions from this IdP. When you make this selection, the associated Value drop-down list is populated by the attribute contract (see *Define an attribute contract* on page 330).
- 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. (If the Expression selection is not listed, then the feature is not enabled.
- JDBC/LDAP/Custom

Values are returned from your query. When you make this selection, the Value list is populated by the database columns or LDAP or custom attributes you identified for this data store (see *Choose a database table and columns* on page 336,*Configure an LDAP directory search* on page 338 , or *Select custom data-source fields* on page 340).

- 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 (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 incoming token assertion, 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 data store attributes you select.

  Tip: Two other 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; this variable is available only if specified as a query parameter for the relevant PingFederate endpoint (either as `TargetResource` for SAML 2.0 or `TARGET` for SAML 1.x—see *Application endpoints* on page 442).

  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 this partner.

To reach this screen:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **User-Session Creation** under the Browser SSO tab.
6. Click **Configure User-Session Creation**.
7. Click **Target Session Mapping** under the User-Session Creation tab.
8. Click the target session name.
9. Click **(Adapter) Contract Fulfillment**on the Summary screen.

To map attributes:

1. Choose a Source for each Target attribute (see descriptions of each Source type above).
2. Choose (or enter) a Value for each Attribute.

   All values must be mapped.
3. Click **Done**.

Identify issuance criteria (optional)

Use this screen to define criteria PingFederate can evaluate to determine whether the user is authorized to continue the SSO transaction using the SP adapter or connect mapping contract (see *About token authorization* on page 71). This token authorization can be used to restrict who can access protected resources.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Assertion – Select to access attributes from the assertion.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for data stores of the same type

   - Mapped Attributes – Select to access the required attributes.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Error results are handled in one of two ways:

   - **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 `sp.sso.error.page.template.html` template (for more information on this and other user-facing templates, see *Customizable user-facing screens* on page 120).

     Note: Using an error code in the Error Result field allows the error template or a web application to process the error result in a variety of ways—for example, a redirect to another page or e-mail to an administrator.

   If you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.
8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Enable and disable expressions* on page 485).

   Important: When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), consider using an OGNL expression to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access (see *Sample OGNL expressions* on page 487).

   - Use the in-line editor box to enter the OGNL expression.

     For more information about OGNL, see *Attribute mapping expressions* on page 485.
   - Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

- Click **Add**. Continue to add expressions, as needed.
- Click the **Test** link to input values and test the resulting output for the expression.

Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

Review the target session mapping

When you have finished adding a new or modifying existing instance of an SP Adapter or an 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.

- If you are editing an existing configuration, click **Done**; if you want to keep your changes, click **Save** when you reach the Browser SSO screen.

Review the session creation summary

- On the Session Creation Summary screen, click any heading to change the configuration; or click **Done**.

  If you are editing an existing configuration, be sure to click **Save** when you return to the Browser SSO screen, if you are finished.

Configure OAuth attribute mapping

This configuration allows administrators to use assertion attributes for mapping values into OAuth persistent-grant `USER_KEY` and extended attributes.

1. On the **Connection Type** screen, select the **Browser SSO Profiles** check box and the applicable protocol.
2. 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 SAML Grant** configuration.
3. Follow the wizard to complete the Browser SSO configuration. On the **OAuth Attribute Mapping** screen, click **Configure OAuth Attribute Mapping** to continue.

Choose an OAuth data store

This optional configuration is the same for all OAuth attribute-mapping task flows.

- If you do not need additional attributes from a data store, just click **Next** on the Data Store screen.

To reach this screen for editing

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.

4. Click **Configure Browser SSO**.
5. Click **OAuth Attribute Mapping** under the Browser SSO tab.
6. Click **Configure OAuth Attribute Mapping**.
7. Click **Data Store** on the Summary screen.

Configure contract fulfillment

The last step in configuring OAuth attribute mapping is to map attributes for the `USER_KEY` (to look up persistent OAuth grants), the extended attributes (to fulfill the access token), and `USER_NAME` (the name shown to end-users on permissions screens) for persistent access-token grants.

Note: The `USER_KEY` values must be unique across all end users because it is the identifier to store and to retrieve persistent grants. If SAML_SUBJECT uniquely identifies all end users, you can map SAML_SUBJECT to `USER_KEY`.

Map each attribute from one of the following Sources:

- Assertion

  Values are contained in the assertions from this IdP. When you make this selection, the associated Value drop-down list is populated by the attribute contract (see *Define an attribute contract* on page 330).
- 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. (If the Expression selection is not listed, then the feature is not enabled.
- JDBC/LDAP/Custom

  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 (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 incoming token assertion, 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 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**.

To reach this screen

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **OAuth Attribute Mapping** under the Browser SSO tab.
6. Click **Configure OAuth Attribute Mapping**.

7. Click **Contract Fulfillment** on the Summary screen.

Specify issuance criteria for OAuth attribute mapping

Use the **Issuance Criteria** screen to 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 a resource.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Assertion – Select to access attributes from the assertion.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for data stores of the same type

   - Mapped Attributes – Select to access the required attributes.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in the value of this field being used by the `error_description` protocol field.

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.
8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear.

   - Use the in-line editor box to enter the OGNL expression.
   - Use the Error Result box to enter an error message or an error code for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the **Manage Mappings** screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

Click **Delete** under Actions for the criteria and then click **Save**.

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 protocol settings

The Protocol Settings screen provides the launching point for configuring bindings, partner endpoints, and other settings needed for the selected SAML profiles (if you are using SAML 2.0—see *Select SAML 2.0 profiles* on page 327). The screen also displays configured information.

(For WS-Federation, the configuration of bindings is not applicable.)

To configure Protocol Settings, you need to know:

- For SP-initiated SSO profiles, the URL(s) of your IdP's Single Sign-on Service(s).
- For SLO profiles, the URL(s) of your IdP's Single Logout Service(s)
- When artifact is an allowable inbound binding, the URL of your IdP's Artifact Resolution Service(s)
- The transport configurations (bindings) that you will use to send and receive data for SSO/SLO connections
- Digital signature policies and certification requirements to which you and your connection partner have agreed
- XML encryption policies to which you and your connection partner have agreed
- To continue, click **Configure Protocol Settings**.

Note: After modifying any settings, you must click **Save** on the Protocol Settings screen.

Specify SSO service URLs (SAML)

At this step for SAML 2.0 connections, you associate bindings to the endpoints where your IdP 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.

This configuration applies only to the SP-initiated SSO Profile (see *About SP-Initiated SSO* on page 327).

Field descriptions

Field	Description
Binding (SAML 2.0)	The transport type agreed upon by you and your partner: Artifact, POST, or Redirect.
Endpoint URL	The location where your IdP receives SSO messages.
MessageType	The type of the message that you need to customize for this connection.
Expression	The OGNL expression to fulfill your use case.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.

2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **SSO Service URLs** on the Summary screen.

   If this step is not displayed, you have not selected SP-initiated SSO (see *Select SAML 2.0 profiles* on page 327).

To define an Endpoint URL:

1. If you are using SAML 2.0, select the Binding your partner specifies for the Endpoint.
2. Enter the fully qualified Endpoint URL or a relative path if you have defined a base URL (see *Identify the IdP* on page 324).
3. If you are using SAML 2.0, click **Add**.
4. If your partner has additional SSO endpoints established under SAML 2.0, repeat the steps above.

To customize the authentication requests:

1. Click **Show Advanced Customizations**.

   Note: Expressions must be enabled for the **Show Advanced Customizations** button to appear (see *Enable and disable expressions* on page 485).
2. Select a Message Type.
3. 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 489.

Specify a service URL (WS-Federation)

The Service URL is the WS-Federation endpoint of your IdP partner. This endpoint is where you send 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 159).

- Enter the fully qualified URL or a relative path if you have defined a base URL (see *Identify the IdP* on page 324).

To define a service URL:

- Enter the fully qualified URL or a relative path if you have defined a base URL (see *Identify the IdP* on page 324).You must include the initial slash if you are entering only a relative path.

To specify additional allowed domains and paths:

1. Indicate whether to 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.
2. Enter the expected domain or IP address under Valid Domain Name.

   Use the domain only, without qualifiers. For example:

`company.com`

Using an initial wildcard and period for a domain name will cover multiple subdomains. For example: `*.company.com`

covers `hr.company.com` or `email.company.com`.

(Optional) Enter the exact path of the resource (case-sensitive) 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. For example:

`/app/Consumer.jsp`

allows `/app/Consumer.jsp` but rejects `/app/consumer.jsp`.

Tip: You can also enter multiple query parameters with or without a fragment.

For example: `/app/Consumer.jsp?area=West&team=IT#ref1001`

matches `/app/Consumer.jsp?area=West&team=IT#ref1001` but not `/app/Consumer.jsp?area=East&team=IT#ref1001`

(Optional) Select the Allow Any Query / Fragment check box if you want to allow any query parameters or fragment in the resource.

Note: When selected, no query parameter and/or fragment is allowed in the path.

Click **Add**.

3. Repeat the previous step to add additional entries as needed.
4. Click **Save**.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** under the Browser SSO tab.
6. Click **Configure Protocol Settings**.
7. Click **Service URL** on the Summary screen.

Define SLO service URLs (SAML 2.0)

At this step you associate bindings to the endpoints where your IdP receives logout requests when SLO is initiated at your site and where you send SLO responses when you receive SLO requests from the IdP.

This step applies only for SAML 2.0 connections when you have selected an SLO profile (see *Select SAML 2.0 profiles* on page 327).

Field descriptions

Field	Description
Binding	The transport type agreed upon by you and your partner: Artifact, POST, Redirect, or SOAP.
Endpoint URL	A location where your IdP receives SLO request messages.
Response URL	(Optional) A location where the IdP receives SLO logout response messages. Use this endpoint when you are part of a chain of session participants. When omitted, the PingFederate SP server sends logout responses to the Endpoint URL.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **SLO Service** on the Summary screen.

To define an Endpoint URL:

1. Select the Binding your partner specifies for the Endpoint.
2. Enter the fully qualified Endpoint URL or a relative path if you have defined a base URL (see *Identify the IdP* on page 324).
3. Optional: Enter the Response URL or a relative path and click **Add**.
4. If your partner provides additional endpoints for SLO, repeat the steps above.

Select allowable SAML bindings (SAML)

At this step you configure binding(s) that the IdP will use to send SAML assertions or SLO messages (under SAML 2.0) to your PingFederate server.

This configuration applies to all profile types (see *Select SAML 2.0 profiles* on page 327). You and your partner can agree to standardize on one binding type or select different bindings for different profile scenarios.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Allowable SAML Bindings** on the Summary screen.

To define binding requirements for this connection:

- Make your selections and click **Next** (or **Done**).

Specify an artifact lifetime (SAML)

When you send an artifact to your IdP's SSO or SLO service , an element in the message indicates how long it should be considered valid.

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.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.

3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Artifact Lifetime** on the Summary screen.

   This step appears only if you have selected the artifact binding for either a SSO or SLO Service at the IdP site.

Define artifact resolver locations (SAML 2.0)

This endpoint or group of endpoints is where your server will send back-channel requests based on artifacts. The location or locations are also known under SAML specifications as the Artifact Resolution Service. SAML 2.0 provides for multiple, indexed endpoints for the service.

Note that this screen is different for SAML 1.x implementations, for which only one endpoint is allowed.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** on the Summary screen.
6. Click **Configure Protocol Settings**.
7. Click **Artifact Resolver Locations** on the Summary screen.

   If this step does not appear, you do not have Artifact selected under **Allowable SAML Bindings**.

For a SAML 2.0 connection:

1. Enter a URL on the Artifact Resolver Locations screen and click **Add**.

   The URL must be fully qualified (defining protocol, host, and port) unless you have entered a base URL (see *Identify the IdP* on page 324).

   Repeat this step if your IdP supports multiple services. The SAML 2.0 specifications permit multiple artifact resolution services through the use of Index numbers, which PingFederate automatically supplies when you add a service. Alternatively, if needed per partner specifications, you may assign these index numbers manually.

   Note: When specifying multiple artifact resolution endpoints, each endpoint must share the same transport protocol. That is, if one endpoint uses HTTP, then all must use HTTP. Similarly, if one endpoint uses HTTPS, then all must use HTTPS.
2. Click **Next**.

For a SAML 1.x connection

1. Enter the Endpoint on the Artifact Resolution Location screen.

   The URL must be fully qualified (defining transport protocol, host, and port) unless you have entered a base URL (see *Identify the IdP* on page 324).
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 SP 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.
3. Click **Next**.

Configure default target URLs (optional)

Use this screen to assign a default target URL for this IdP Connection configuration. Entering a URL in the Default Target URL field overrides any SP Default URL SSO setting (see *Configure default URLs* on page 315).

Note: 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 (see *IdP endpoints* on page 442), any URL specified in the Default Target URL field on this screen will not be used for those transactions.

Configure signature policy

The Signature Policy screen provides options controlling how digital signatures are used for SSO Internet messaging. The choices made on this screen depend on your partner agreement (see *Digital signing policy coordination* on page 74).

Digital signing is required for SAML Response messages sent from the IdP via POST (or Redirect for SAML 2.0). Optionally, SSO authentication requests from the SP (SP-initiated SSO) may also be signed to enforce security. (This option appears only for SAML 2.0 connections and only if you have enabled SP-initiated SSO using the POST or redirect bindings.)

The assertions inside SAML Responses may be also be signed. When you make this choice, only the assertion portion of the Response is signed, not the complete Response. (This is the only option that appears for SAML 1.x connections.)

- To choose one or more enhancement options, select the second button, make your selection(s), and click **Next**.
- Otherwise, select the first option (if not already selected) and click **Next**.

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. This feature is commonly used if the assertion might pass through an intermediary (such as a user's browser) and HTTPS is not used.

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 single-logout 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 reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Browser SSO** under the IdP Connection tab.
4. Click **Configure Browser SSO**.
5. Click **Protocol Settings** under the Browser SSO tab.
6. Click **Configure Protocol Settings**.
7. Click **Encryption Policy** on the Summary screen.

To define XML encryption:

1. Select Allow encrypted SAML Assertions and SLO messages.
2. Choose whether this IdP partner will encrypt the entire assertion, the `SAML_SUBJECT`, one or more other attributes, or some combination.
3. If your partner is encrypting the name-identifier attribute, use the check boxes near the bottom of the screen to indicate whether you will encrypt this attribute in outbound SLO messages and/or allow its encryption for inbound messages.
4. Click **Next** or **Done**.

To disable previously configured XML encryption selections:

1. Select **None** and then **Done**.
2. Click **Save** on the Browser SSO screen.

Edit and save protocol settings

On the Summary screen, you can review or edit your Protocol Settings.

Important: When you finish editing existing settings, be sure to click **Done** on the Summary screen and then **Save** on the Protocol Settings screen. For a new connection, click **Done** and then click **Next** on the Protocol Settings screen. Save the entire connection on the Activation & Summary screen (see *Review IdP Auto-Connect settings* on page 377).

To reconfigure saved settings:

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 dependent or other changes, do so and continue by clicking **Done** until you reach the Protocol Settings screen.
3. Click **Save** on the Protocol Settings screen.

Review Browser SSO settings

On the Summary screen for Browser SSO, you can review or edit your SSO configuration.

Important: When you finish editing existing settings, be sure to click **Done** on the Summary screen and then **Save** on the Browser SSO screen. For a new connection, click **Done** and then click **Next** on the Browser SSO screen. Save the entire connection on the Activation & Summary screen (see *Review an IdP connection* on page 375).

To reconfigure saved settings:

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 dependent or other changes, do so and continue by clicking **Done** until you reach the Browser SSO screen.
3. Click **Save** on the Browser SSO screen.

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 322). 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* in the “Supported Standards” chapter of the *Get started with PingFederate* guide).

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* in the Supported Standards chapter of the *Get started with PingFederate* guide).

To configure the URL:

- Enter the fully qualified URL or a relative path if you have defined a base URL (see *Identify the IdP* on page 324).

Mapp 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 369).

This screen also allows you to mask incoming attribute values in log files (see *Attribute masking* on page 71). 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 from IdPs (see *Just-in-time provisioning*). 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 (see *Choose IdP connection options*).

- To continue, click **Configure User 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 (see *Attribute Query and XASP* and *Choose IdP connection options*). The **User Attributes** screen displays the attributes expected in the assertion from this IdP (see *Define an attribute contract*).

Note: Attribute Query is a SAML 2.0 profile. For 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, if you have not already done so (see *Manage Attribute Query profile*).

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.

- Choose the Active Data Store on the User Repository screen.

  If the correct data store is not shown in the drop-down list, then PingFederate is not yet configured to access the store (see *Manage data stores*).
- If you are using an LDAP store, refer to the sections immediately following:
  - *Specify an LDAP user-record location* on page 354
  - *Enter an LDAP filter* on page 355
  - *Identify provisioning attributes for LDAP* on page 355
- If you are using MSSQL, 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.

Note: The **Location** screen appears only for LDAP data stores (see *Identify the user repository* on page 354).

Field description

Field	Description
Base DN	The base distinguished name 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 (see the previous section) to locate existing account records and to add new ones.

Note: This screen appears only for LDAP data stores (see *Identify the user repository* on page 354).

The filter is in the form: `attribute=${value}`

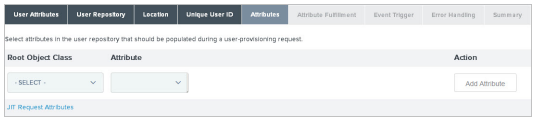
Note that the statement must not be enclosed in parentheses, unlike filters used to retrieve LDAP attributes for adapter mapping (see *Define an LDAP filter*).

The left-side variable is an attribute in your user-data store—click the link near the 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 assertions (see *Define an Attribute Contract*). Variables for these attributes, including the correct syntax, are listed under Assertion Values.

Note: 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.



Note: This screen appears only for LDAP data stores (see *Identify the user repository* on page 354).

1. Choose a Root Object Class and an Attribute from the drop-down lists and 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.

Note: The SQL Method screen appears only for JDBC data stores (see *Identify the user repository* on page 354).

- Make a selection as needed and click **Next**.

  Depending on the selection, different steps appear under the JIT Provisioning task. Refer to the manual 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 356
  - *Specify a unique-ID database column* on page 356
- If you are using a Stored Procedure, skip to this section:
  - *Specify a stored-procedure location* on page 356

Specify a database user-record location

For database provisioning to a table, indicate where PingFederate should write new user records or update existing ones.

Note: This screen appears only for MSSQL data stores when Table is selected on the SQL Method screen (see the previous section, *Choose a SQL method* on page 355).

Field descriptions

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 when provisioning	For the selected table, all attributes and their data types are displayed. All attributes must be mapped for the database insertion to succeed, although a null entry may be used for optional attributes (see *Map attributes to a user account* on page 356).

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 assertion.

Note: This screen appears only when Table is selected on the SQL Method screen (see *Choose a SQL method* on page 355).

- Select a column that represents a unique characteristic about the database entry for a particular user (email, for example).

Specify a stored-procedure location

If you are using a stored procedure for provisioning the user database, specify its location on this screen.

Important: The database account used by PingFederate must have access to the schema in which the stored procedure is located, as well as execute permission for the procedure.

Note: This screen appears only when Stored Procedure is selected on the SQL Method screen (see *Choose a SQL method* on page 355).

Field descriptions

Field	Description
Schema	Select the table structure that contains stored procedures 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. You map assertion values to the parameters on the next screen.

- To see a list of attributes expected from SAML assertions, click **View Attribute Contract**.
- If the list of parameters is not or might not be current (due to any recent procedure revisions), click **Refresh**.

Map attributes to a user account

The **Attribute Fulfillment** screen provides a means of mapping the values of incoming attributes into local account attributes or, for JDBC stores, into SQL stored-procedure parameter values (see *Choose a SQL method*). You can also provide values of your own for any data store attributes (except JDBC system-managed) or for SQL procedure parameters—either as hard-coded text or derived values based on assertion attributes.

For JDBC, this screen also provides a means of testing the insertion of attribute values into the database or stored procedure.

Tip: For mapping to a database `datetime` or `smalldatetime` column, 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 (see *Attribute mapping expressions*). Note that expressions must be enabled to use the PingFederate conversion method—see *Enable and disable expressions*.

Note: For a JDBC data store, if stored procedures are used, the note under the task bar on this screen is different than shown above and the Target Attribute column reads "Target Parameter."

Map attributes from one of these Sources:

- Assertion

  Values are contained in the assertions from this IdP. When you make this selection, the associated Value drop-down list is populated by the attribute contract (see *Define attribute contract*).
- Context

  Values are returned from the context of the transaction at runtime—for example, authentication context.
- Attribute Query

  This choice appears only if you have chosen to use the Attribute Query profile for provisioning (see *Select attribute sources (SAML 2.0)* on page 354).

  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 (when enabled—see *Enable and disable expressions*)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions*). All of the variables available for text entries (see below) are also available for expressions.

  Tip: If multiple attribute values from one or multiple sources need to be mapped to one LDAP attribute value, use an OGNL expression to create it.

  Tip: For JDBC mapping, if the data type of a Target Attribute/Parameter is `datetime` or `smalldatetime`, you can use an expression to convert date-time strings from the assertion. 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 JDBC attributes are among columns to be provisioned—for example, an identity or timestamp column for the MSSQL 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 assertion, using the `${attribute}` syntax.

  Note: If no entry is required in a JDBC database 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 table column.

To map attributes:

1. Choose a Source for provisioning each Target Attribute or Target Parameter (see descriptions of each Source type above).

   Note: For JDBC table mapping, select System Managed as the Source for any columns that are automatically provisioned by the database server.

   Tip: For LDAP mapping, choose Text as the Source for the `objectClass` attribute.

2. Choose (or enter) a Value for each Attribute.

   All values must be mapped. However, for optional JDBC 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`.

To test the insertion of attributes into a JDBC table or stored procedure:

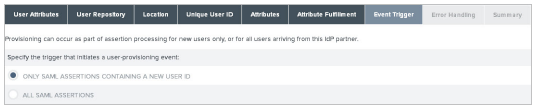
1. Click the **Test insert into . . . or Test call to . . .** link.
2. For each Target Attribute or Target Parameter (for a stored procedure), enter text into the boxes under Test Value and click **Test Insert** (or **Test Stored Procedure Call**).

   For table insertions, if the test is successful, a confirmation is displayed along with the values inserted. For stored procedures, only a confirmation is displayed if the test is successful, indicating that the procedure was populated with parameter values.
3. For table insertions, unless you wish to keep test values in the database, click **Roll Back All Test Inserts**.

   If you are using a stored procedure, this option is not provided because PingFederate does not know the result of the procedure. Database rollback, if needed, must be handled manually.
4. When finished, click **Next** or **Done**; or click the link **Return to Attribute Fulfillment** to continue or change any mapping.

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 a SAML assertion. In the latter case (for all assertions), an existing user account is always updated with incoming attributes.



Note: This screen does not appear for JDBC data stores if provisioning is accomplished using a stored procedure (see *Choose a SQL method*), because the procedure is always called for all assertions. 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 assertion attributes on to your application (via the SP adapter configuration—see *Manage target session mappings*).

When SSO is aborted, the user is redirected to an error page and the failure is written to the log.

Review the JIT provisioning configuration

The **Summary** screen provides an overview of your provisioning configuration.

- When you are finished with a new configuration, click **Done** and then **Save** on the **JIT Provisioning** screen.

To change the 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** on the JIT Provisioning screen.

Configure inbound provisioning

Inbound Provisioning implements the SCIM protocol to provide an automated user-management service when PingFederate is configured as an SP (see *Provisioning for SPs* on page 79).

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 363). 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 364).

The dual mapping is intended to provide greater flexibility, especially when needed for OGNL-expression transformations—for example, converting two attributes into one multi-value attribute and then back again (see *Attribute mapping expressions* on page 485).

Note: SCIM-client requests must include authentication credentials, which you configure later (see *Configure security credentials* on page 369). 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.

- To continue, click **Configure Inbound Provisioning** in the Inbound Provisioning screen.

Specify the user repository

PingFederate currently supports AD user stores (requires an LDAPS connection to the data store) as well as custom identity store provisioners for Inbound Provisioning.

- Choose either Active Directory Data Store or Identity Store Provisioner and then specify the data store from the drop-down menu.

  Note: If the correct data store is not shown in the drop-down list, then PingFederate is not yet configured to access the store (see *Manage data stores* on page 143).

  If the identity store provisioner is not shown in the drop-down list, then PingFederate is not yet configured to access the store (see *Configure Identity Store Provisioners* on page 313).

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.

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

To specify a base DN:

- Enter the base Distinguished Name of the tree structure where user records are stored. 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 (see the previous section) to add new account records.

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

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 (see *Define an LDAP filter* on page 339).

The left-side variable is an attribute in your user-data store—click the link near the 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.

Note: If you are unfamiliar with writing LDAP queries, please refer to the documentation accompanying your LDAP installation.

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

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 (see the previous section) to add new groups.

Note: This screen appears only if you are configuring an LDAP user store for provisioning and the User and Group Support check box is selected in the Connection Type screen (see *Choose an IdP connection type* on page 322).

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 (see *Define an LDAP filter* on page 339).

The left-side variable is an attribute in your user-data store—click the link near the 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.

Note: If you are unfamiliar with writing LDAP queries, please refer to the documentation accompanying your LDAP installation.

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

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** in 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 Multi-Valued Attributes
- Complex Attributes
- Complex Multi-Valued 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 Multi-Valued	Not Applicable
`options`	Complex	`quantity`, `strike`, `first`, and `last`
`tablets`	Complex Multi-Valued	`model`, `serial`, and `type`.

Note: `type` is a reserved sub-attribute for a complex multi-valued attribute.

Tip: For more information about SCIM and attribute types, see the web site *www.simplecloud.info*.

To specify a schema extension:

- Specify the URI of the schema extension in the Extension namespace field.

  Tip: The default value is `urn:scim:schemas:extension:custom:1.0`. You can keep this value if your partner identifies custom attributes by this URI in its SCIM messages.

To add a custom attribute:

- Enter an attribute name and click **Add**. (Repeat this step to add more custom attributes as needed.)

To delete a custom attribute:

- Click **Delete** next to the custom attribute. (To undo the deletion, click **Undelete**.)

To edit a custom attribute:

- Click **Edit** next to the custom attribute to:
  - Change its attribute name
  - Set it as a simple multi-valued attribute
  - Add sub-attributes to make it a complex attribute
  - Add sub-attributes and set it as a complex multi-valued attribute

  (For more information, see *Configure custom SCIM attribute options* on page 362.)

When you finish editing custom attributes, click **Next** in the Custom SCIM Attributes screen.

Configure custom SCIM attribute options

For the chosen custom attribute, use the Custom SCIM Attribute Options screen to:

- Change its attribute name
- Set it as a simple multi-valued attribute
- Add sub-attributes to make it a complex attribute
- Add sub-attributes and set it as a complex multi-valued attribute

To change the name of the custom attribute:

1. Replace the current value in the Name field.
2. Click **Done**.

To define the custom attribute as a simple multi-valued attribute:

1. Select the **Is Multi-Valued** check box.
2. Click **Done**.

To define the custom attribute as a complex attribute:

1. Enter a sub-attribute and click **Add**. (Repeat this step to add more sub-attributes as needed.)

   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.
2. Click **Done**.

To define the custom attribute as a complex multi-valued attribute:

1. Enter a sub-attribute and click **Add**. (Repeat this step to add more sub-attributes as needed.)

   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.
2. Select the **Is Multi-Valued** check box.
3. If you have chosen Active Directory as your user store (see *Specify the user repository* on page 359), you must specify at least one value under the Types column for `type`, a reserved sub-attribute for a complex multi-valued attribute.

   Important: If an inbound SCIM message includes this complex multi-valued attribute with multiple entries, each entry must contain the `type` sub-attribute with a unique value matching one of the values specified under the Types column. Work with you partner to identify the possible `type` values and enter them here.

   Note: If you have chose Identity Store Provisioner as your user store, the `type` sub-attribute is optional.

   Tip: Use the **Edit**/**Update**/**Cancel** links to make or undo a change to the `type` value. Use the **Delete**/**Undelete** links to remove a `type` value or cancel the deletion.
4. Click **Done**.

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.

- 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.

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

Select system-managed LDAP attributes

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.

To select attributes:

1. Choose a Root Object Class and an Attribute from the drop-down lists and click **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.

Map attributes from one of these sources:

- 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. (If the Expression selection is not listed, then the feature is not enabled.

- Expression (when enabled—see *Enable and disable expressions* on page 485)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 485). 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.

To map attributes:

1. Choose a Source for provisioning each Target Attribute (see descriptions of each Source type above).
2. Choose (or enter) a Value for each Attribute.

   All values 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**.

To change the 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 Writer Users screen.
3. Click **Next** on the Inbound Provisioning screen.

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.

- 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 at the top of the screen.

Use the Available SCIM Attributes link at the bottom 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.

Select system-managed SCIM attributes

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:

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. To see a list of available attributes, click **Available SCIM attributes**.
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 **Cancel** 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.

Identify LDAP attributes for the SCIM response

Select the LDAP attributes you want to map to attributes in the SCIM response.

Note: This screen appears only if you are configuring an LDAP user store for provisioning.

To select attributes:

1. Choose a Root Object Class and an Attribute from the drop-down lists and 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 attributes from one of these sources:

- 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. (If the Expression selection is not listed, then the feature is not enabled.

- Expression (when enabled—see *Enable and disable expressions* on page 485)

  This option provides more complex mapping capabilities—for example, transforming outgoing values into different formats (see *Attribute mapping expressions* on page 485). 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.

To map attributes:

1. Choose a Source for provisioning each Target Attribute (see descriptions of each Source type above).
2. Choose (or enter) a Value for each Attribute.

   All values 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**.

To change the 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 Read Users screen.
3. Click **Save** on the Inbound Provisioning screen.

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 (see *Choose an IdP connection type* on page 322), when PingFederate receives a SCIM delete request for a group, it always removes the specified group from the data store.

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 protocol*).

  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.

- 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.

Note: This screen appears only if you are configuring an LDAP user store for provisioning and the User and Group Support check box is selected in the Connection Type screen (see *Choose an IdP connection type* on page 322).

Select system-managed LDAP group attributes

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.

To select attributes:

1. Choose a Root Object Class and an Attribute from the drop-down lists and 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 attributes from one of these sources:

- 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. (If the Expression selection is not listed, then the feature is not enabled.

- Expression (when enabled—see *Enable and disable expressions* on page 485)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 485). 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.

To map attributes:

1. Choose a Source for provisioning each Target Attribute (see descriptions of each Source type above).
2. Choose (or enter) a Value for each Attribute.

   All values 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**.

To change the 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 Write Groups screen.
3. Click **Next** on the Inbound Provisioning screen.

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.

- 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.

Use the Available SCIM Attributes link at the bottom 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.

Select system-managed SCIM group attributes

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:

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. To see a list of available attributes, click **Available SCIM attributes**.
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 **Cancel** 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.

Identify LDAP group attributes for the SCIM response

Select the LDAP attributes you want to map to attributes in the SCIM response.

Note: This screen appears only if you are configuring an LDAP user store for provisioning and the User and Group Support check box is selected in the Connection Type screen (see *Choose an IdP connection type* on page 322).

To select attributes:

1. Choose a Root Object Class and an Attribute from the drop-down lists and 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 attributes from one of these sources:

- 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. (If the Expression selection is not listed, then the feature is not enabled.
- Expression (when enabled—see *Enable and disable expressions* on page 485)

This option provides more complex mapping capabilities—for example, transforming outgoing values into different formats (see *Attribute mapping expressions* on page 485). 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.

To map attributes:

1. Choose a Source for provisioning each Target Attribute (see descriptions of each Source type above).
2. Choose (or enter) a Value for each Attribute.

   All values 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**.

To change the 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 Read Groups screen.
3. Click **Save** on the Inbound Provisioning screen.

Review the inbound provisioning configuration

On the Summary screen you can review the Inbound Provisioning 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 Inbound Provisioning screen.
3. Click **Save** on the Inbound Provisioning screen.

Configure security credentials

The Credentials screen presents a list of possible 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 370
- *Manage digital signature settings* on page 371
- *Manage signature verification settings* on page 372
- *Choose an encryption certificate (SAML 2.0)* on page 374
- *Choose a decryption key (SAML 2.0)* on page 375

- To configure or modify credentials, 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.

This step also applies to attribute-request configurations, because this profile always uses the SOAP back channel (see *Select SAML 2.0 profiles* on page 327).

Note: A yellow triangle next to a listing indicates that you have not completely configured back-channel authentication requirements.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.

   If the Back-Channel Authentication step is not shown, then it is not applicable to your configuration—you are not using the Attribute Query profile and have not configured any profiles to use the artifact or SOAP bindings.

To configure back-channel authentication requirements for sending HTTP messages:

1. On the Back-Channel Authentication screen, click the **Configure** link to the right of the list of messages to be *sent* to your partner.
2. Make one or more selections on the Outbound SOAP Authentication Type screen:
   - HTTP Basic — you enter SOAP Basic credentials on a later screen.
   - SSL Client Certificate — you specify the certificate on a later screen.

     This option is available only if you specify an endpoint that uses SSL.
   - Use Digital Signatures (Browser SSO profile only) — you sign the message.

     You are asked to select a signing certificate on a later screen.

   For SAML 2.0, these options may be used in any combination or independently. For SAML 1.x, you must use either Basic or SSL authentication; digital signing may be added to ensure message integrity.

   By default, PingFederate validates your partner's SSL server certificate— verifying that the certificate chain is rooted by a trusted Certificate Authority and that the hostname matches the certificate's Common Name. Clear the associated check box if you do not want this validation to occur.
3. Click **Next**.
4. If you chose HTTP Basic at *Step 2*, enter the SOAP Username and Password to use for this partner under Basic SOAP Authentication.

   You must obtain these credentials from your partner.
5. If you are using an SSL certificate, select the certificate under SSL Authentication Certificate and click **Next**.

If you have not yet created or imported the client SSL certificate you need into PingFederate, click **Manage Certificates** (see *Manage SSL client keys and certificates* on page 163). You need to export the certificate (only) and send it your partner.

6. On the Summary screen, click **Done**.

To configure back-channel authentication requirements for receiving HTTP messages:

1. On the Back-Channel Authentication screen, click the **Configure** link to the right of the list of messages to be *received* from your partner.
2. Select one or more options on the Inbound Authentication Type screen:
   - HTTP Basic — Enter the logon username and password your partner uses on the next screen.
   - SSL Certificate — Specify certificate verification information on a later screen.
   - Use Digital Signatures (Browser SSO profile only). . . — Incoming messages must be signed.
   - Require SSL — When selected, incoming HTTP transmissions must use a secure channel.

   You are asked to select a signature verification certificate on a later screen.

   For SAML 2.0, use these options in any combination or independently. For SAML 1.x, you must use either Basic or SSL authentication; add digital signing to ensure message integrity.
3. Click **Next**.
4. If you chose HTTP Basic at *Step 2*, enter the Username and Password under Basic Authentication (Inbound).

   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.
5. If you are using an SSL certificate, select Anchored or Unanchored under Certificate Verification Method.
   - Anchored — The certificate must be signed by a trusted Certificate Authority. 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 (see *Manage trusted certificate authorities* on page 160).
   - Unanchored — The 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.
6. Click **Next**.
7. If you chose anchored SSL certificate verification at *Step 5* , enter the Subject DN and click **Next**.

   Tip: If you have not yet defined the certificate in PingFederate or you do not know the DN, return to the previous screen and select Unanchored. Then click **Next** and click **Manage Certificates** on the SSL Verification Certificate screen to import the certificate, if needed, or to view its DN.
8. If you chose unanchored SSL certificate verification at *Step 5*, select the certificate to use for validating the SSL connection.

   If you have not yet imported the certificate into PingFederate, click **Manage Certificates**.
9. Click **Next**.
10. On the Summary screen, click **Done**.

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 (see *Select SAML 2.0 profiles* on page 327) 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 (see *Configure security policy for Attribute Query* on page 353).

The step is not required for SAML 1.x IdP connections.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.
5. Click **Digital Signature Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require digital signatures. You do not have SLO configured using the POST or redirect bindings, and you have not elected to sign either authentication or attribute requests (see *Configure signature policy* on page 351 and *Configure security policy for Attribute Query* on page 353).

To specify a certificate:

1. Select the certificate from the drop-down list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing and decryption keys and certificates* on page 164).
2. Optional: If you have agreed to send your public key with the SAML message, select the **Include this certificate's public key certificate ...** check box. To include the raw key in the signature as well, select the **Include the raw key …** check box.
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.

Whenever signatures are required, PingFederate provides a choice of trust models, including an option to use anchored signature-verification certificates embedded in incoming messages (see *Trust Models*). When this option is chosen in Signature Verification Settings, you must provide the Subject DN for embedded certificates coming from this partner, and the Issuer CA certificate must be part of the PingFederate trusted store (see *Manage trusted certificate authorities* on page 160).

Alternatively, you may choose to use unanchored certificates, in which case you must import your partner's public-key certificate during this configuration (or select it if it is already imported). To prevent any interruption of service due to an expired certificate, you can ask your partner for a new certificate in advance and import it as backup.

- To continue, click **Manage Signature Verification Settings**.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.

5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.

Select a trust model

This screen allows you choose the Trust Model you want to use for signature verification (see *Trust models* on page 74).

- Depending on the selection, the next step in this task varies:
  - For **Anchored**, the next step is to enter the Subject DN for your partner's certificate (see the next section, *Define a Subject DN for anchored certificates* on page 373).

    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.
  - For **Unanchored**, the next step is to import your partner's certificate (see *Choose unanchored certificates* on page 374).

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.
7. Click **Trust Model** on the Summary screen.

Define a Subject DN for anchored certificates

When you choose to use an anchored certificate for signature verification, incoming SAML messages must contain the partner's verification certificate (see *Trust Models*). PingFederate verifies that the Issuer DN (if specified) matches that of one of the issuers in the chain, the Issuer CA is trusted and the embedded certificate's Subject DN matches the one specified on this screen. If so, PingFederate uses that certificate to verify the message signature.

To complete the configuration:

1. Enter the Subject DN or extract it from your IdP partner's certificate if the certificate is stored on an accessible file system.

   Tip: To extract the Subject DN from a certificate, browse to select your IdP partner's public certificate and click **Extract**.
2. Optional: Select the Restrict Issuer check box. Enter the Issuer DN or extract it from a certificate if it is stored on an accessible file system.

   Important: We recommend enabling this option to mitigate potential man-in-the-middle attacks and to provide a means to isolate certificates used by different connections.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.

5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.
7. Click **Certificate Subject DN** on the Summary screen.

Choose unanchored certificates

On the Signature Verification Certificate screen, you identify your partner's imported public certificate and, optionally, a secondary certificate to use when the first expires (see *Trust Models*).

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.
5. Click **Signature Verification Settings** on the Summary screen.

   If this step does not appear, then your configuration does not require verification settings.
6. Click **Manage Signature Verification Settings**.
7. Click **Signature Verification Certificate** on the Summary screen.

To specify a verification certificate:

1. Select the certificate from the drop-down list.

   If you have not yet imported the certificate into PingFederate, click **Manage Certificates**.
2. Optional: Select a Secondary certificate for backup.

   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.

Edit and save signature verification settings

- Click any heading to change a setting, or click **Done** if the configuration is complete, then **Done** again on the Signature Verification Settings screen.

  Be sure to click **Save** on the Credentials screen if you are editing an existing connection.

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 351.

To enable this XML encryption, you must identify an encryption certificate for this partner.

You must also choose a certificate if encryption of the Name Identifier is required for an Attribute Request profile (see *Configure security policy for Attribute Query* on page 353).

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.

5. Click **Select XML Encryption Certificate** on the Summary screen.

   If this step is not present, then you have either not configured this connection to use the SP-initiated SLO profile (see *Select SAML 2.0 profiles* on page 327) or you have chosen not to encrypt the assertion or the SAML_SUBJECT (see *Specify XML encryption policy (for SAML 2.0)* on page 351).

To identify the encryption certificate:

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 download web site* (`www.oracle.com/technetwork/java/javase/downloads/index.html`).

For more information about XML block encryption and key transport algorithms, see the *XML Encryption Syntax and ProcessingW3C Recommendation* (`www.w3.org/TR/xmlenc-core/`).

Note: As a 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, we recommend changing the selection for increased security.

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 351).

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click a connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **Credentials** under the IdP Connection tab.
4. Click **Configure Credentials**.
5. Click **Select XML Decryption Key**.

   If this step is not present, you have not chosen to require encryption of all or part of the SAML assertion (see *Specify XML encryption policy (for SAML 2.0)* on page 351).

To identify the decryption key:

1. Select an XML decryption key from the **Primary** list.
2. Optional: Select another XML decryption key from the **Secondary** list.

   If the desired XML decryption key is not in the list, click **Manage Certificates** to create or import it.
3. Click **Next**.

Edit and save credential settings

From the Summary screen you can review or edit your credentials configuration.

Important: When you finish editing existing settings, you must click **Done** on the Summary screen and then **Save** on the Credentials screen. For a new connection, click **Done** and then click **Next** on the Credentials screen. Save the entire connection on the Activation screen (see *Review IdP Auto-Connect settings* on page 377).

Review an IdP connection

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 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 SSO or SLO transactions to or from this partner are disabled, as well as access to the WS-Trust STS for Web Service Providers associated with this connection.

Tip: The SSO Application Endpoint near the top of the Summary screen is an example URL that webmasters or web application developers at your site might use to invoke SSO for the connection. For details about SSO and other server endpoints, including optional query parameters, see *Application endpoints* on page 442.

To change a Connection Status:

- Select either Active or Inactive and then click **Save**.

To modify a connection setting:

1. If you know which step needs to be modified, click its link under the IdP Connection tab.

   If you do not know where to change the setting, locate the currently configured data under one of the summary headings and then click the subheading above the data.
2. Change the information on the step screen and click **Save**, if available.

   If **Save** is not available, you are in the middle of a task (see *Tasks and steps* on page 27 in the *Get started with PingFederate* guide); click **Next** or **Done** until you reach a screen containing a **Save** button. Then click **Save** and continue as needed until you return to the Main Menu.

   If your modification requires related configuration changes, PingFederate provides error messages indicating the necessary steps and then guides you to the related screens (unless you click **Cancel**).

   Important: Be sure to click **Save** whenever that button appears, if you want to keep your changes.

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 136).

Once Auto-Connect is enabled on your PingFederate server, complete the configuration from the Main Menu under SP Configuration. 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 33 in the “Supported Standards” chapter of the *Get started with PingFederate* guide).

Note: The signing certificate is embedded in your server's Auto-Connect metadata (see *Using Auto-Connect* on page 75); 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 141). If you use a different certificate, ensure that it meets Auto-Connect validation requirements (see *Auto-Connect security model* on page 77).

To specify a certificate:

1. Select the certificate from the drop-down list.

   If you have not yet created or imported your certificate into PingFederate, click **Manage Certificates** (see *Manage digital signing and decryption keys and certificates* on page 164).
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 328.

  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 330).

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 141).
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 until you return to the Main Menu.

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 STS provides security-token validation and creation to extend SSO access to identity-enabled Web Services (see *About WS-Trust STS* on page 57).

The chapter provides instructions for configuring the WS-Trust STS, including:.

- *Server settings* on page 378
- *IdP configuration for STS* on page 381
- *SP configuration for STS* on page 402

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 136). 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 379).

Enable the WS-Trust STS

You can enable the WS-Trust STS when you first install PingFederate (see *Start PingFederate for the first time* in the "Installation" chapter of the *Get started with PingFederate* guide). If you have already installed PingFederate or are upgrading to a new version, use the following procedure.

To enable WS-Trust and make the STS available for partner connections:

1. Click **Server Configuration** on the Main Menu.
2. Click **Server Settings** under System Settings.

3. Click **Roles and Protocols** on the Summary screen.
4. Select **WS-Trust** for either the IdP or the SP role, or both, depending on your requirements.

   Note: PingFederate fully supports the STS with or without selections of any of the Browser SSO protocols listed above the WS-Trust selections. SAML 1.1 and 2.0 token handling is independent of supported SSO protocols chosen here.
5. Click **Next**.
6. On the Federation Info screen, ensure all required fields are completed.

   Note: Identifiers are needed 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 return later if needed. (For more information about the fields on this screen, see *Specify federation information* on page 137).
7. Optional: Click **Next** to go to the WS-Trust STS Settings screen (see the next section, *Configure STS authentication* on page 379).
8. Click **Save** (on any screen).

Configure STS authentication

Server settings may be configured to require that client applications provide credentials to access the PingFederate STS. This is recommended for IdP configurations using the Username Token Processor (available separately).

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, administrators may want 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 or, using more fine grained controls, at the connection level via Issuance Criteria (see *Token translator mappings* on page 421).

- To continue, click **Configure WS-Trust STS Authentication**.

Select authentication methods

You can choose either HTTP Basic or mutual SSL/TLS authentication (or both) on the Authentication Methods screen. (Note that if both methods are configured, *all* clients must authenticate using both, not one or the other.)

Important: If you choose mutual SSL/TLS authentication, you must configure a secondary PingFederate SSL port (see the property `pf.secondary.https.port` in the table under *PingFederate properties* on page 112).

Configure basic authentication

For HTTP Basic authentication, create username/password pairs (“Users”) for all client applications needing access to the STS.

On the HTTP Basic Authentication screen, you can also delete users and update account passwords.

To add users:

1. Click **Create User**.
2. On the User Account screen, enter a Username and Password, and confirm the password.

   Passwords must be at least six characters long, containing at least one uppercase, one lowercase, and one numeric character.
3. Click **Done**.
4. Repeat the preceding steps as needed.
5. On the HTTP Basic Authentication screen, click **Next**.

   (If you are also configuring SSL authentication, complete that configuration and click **Next** to reach the Summary screen (see *Configure mutual SSL authentication* on page 380).)

6. On the Summary screen, click **Done**.
7. On the WS-Trust STS Settings screen, click **Save**.

To update an account password:

1. Click the Username.
2. On the User Account screen, enter the Current User Password and a New Password, with confirmation.

   Passwords must be at least six characters long, containing at least one uppercase, one lowercase, and one numeric character.
3. Click **Done**.
4. On the HTTP Basic Authentication screen, click **Done**.
5. On the WS-Trust STS Settings screen, click **Save**.

To delete a user:

1. Click **Delete** under Action for the Username.
2. Click **Done** (or **Next** for new configuration).
3. Click **Save** when you reach the WS-Trust STS Settings screen.

Configure mutual SSL authentication

When SSL authentication is selected on the Authentication Methods screen, the configuration begins on the Mutual SSL Authentication screen.

Important: If you choose mutual SSL/TLS authentication, you must configure a secondary PingFederate SSL port (see the property `pf.secondary.https.port` in the table under *PingFederate properties* on page 112).

- To continue, click **Configure Mutual SSL Authentication**.

Choose certificate authentication options

On the Authentication Options screen, select whether to verify client SSL certificates against a list of Subject Distinguished Names (DNs), a list of issuer public certificates imported into PingFederate, or both.

Note: When both options are selected, they are not used alternatively at runtime; both validations are applied.

- To continue, select one or both options and click **Next**.

Manage allowed Subject DNs

On the Allowed Subject DNs screen you can add, edit, or delete Subject DNs for clients allowed to access the PingFederate STS.

To add DNs:

1. Enter a valid Subject DN for a partner STS client and click **Add**.
2. Add other DNs as needed.
3. For a new configuration, click **Next** or **Done** to continue.
4. If you are finished with a new or existing configuration, continue clicking **Done** until you reach the WS-Trust STS Settings screen and then click **Save**.

To edit DNs:

1. Click **Edit** under Action for the Subject DN.
2. Make changes and click **Update**.
3. Click **Done**.
4. If you are finished with a new or existing configuration, continue clicking **Done** until you reach the WS-Trust STS Settings screen and then click **Save**.

To delete entries:

1. Click **Delete** under Action for the Subject DN.
2. Click **Done**.
3. If you are finished with a new or existing configuration, continue clicking **Done** until you reach the WS-Trust STS Settings screen and then click **Save**.

Manage allowed certificate issuers

When STS access is restricted by issuer certificate, the Allowed Issuer Certificates screen provides a means of maintaining a list of valid certificates.

On this screen you can add or remove certificates.

To add certificates:

1. Select the certificate from the drop-down list and click **Add**.

   If the certificate you are looking for is not in the list, click **Manage Certificates** to import it from your file system.
2. Add other certificates as needed.
3. For a new configuration, click **Next** or **Done** to continue.
4. If you are finished with a new or existing configuration, continue clicking **Done** until you reach the WS-Trust STS Settings screen and then click **Save**.

To delete a certificate from the list:

1. Click **Remove** under Action for the Issuer Certificate.
2. Click **Done**.
3. If you are finished with a new or existing configuration, continue clicking **Done** until you reach the WS-Trust STS Settings screen and then click **Save**.

Review the configuration for mutual SSL authentication

When you have finished configuring Mutual SSL Authentication, 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.

- To save a new or modified configuration, click **Done** on successive screens until you reach the WS-Trust STS Settings screen and then click **Save**.

Review the STS configuration

When you have finished configuring WS-Trust STS 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.

- If you are editing an existing connection, click **Done** and on the WS-Trust STS Settings screen click **Save**.

IdP configuration for STS

This section covers the IdP configuration for the PingFederate WS-Trust STS, which involves:

- *Manage token processors* on page 381
- *Manage STS request parameters* on page 386 (Optional)
- *Configure SP connections for STS* on page 386

Manage token processors

Token Processors are used to validate incoming tokens and token requests to the STS (see *Token processors and generators* on page 58). Token Processors for SAML, OAuth, JWT tokens, and username tokens are included with the PingFederate installation. This section provides guidance on configuring “instances” of these installed Token Processors.

You must configure at least one processor in order to set up an STS connection or token-to-token mapping (see *Token translator mappings* on page 421).

Important: If more than one instance of the same token-processor type is configured for a connection or token-to-token mapping, clients calling either of the PingFederate STS endpoints must add a query parameter, `TokenProcessorId`, and specify the Instance Id. (For endpoint information, see *View IdP protocol endpoints* on page 244.)

For example:

```
https://<pf_host>:<pf_port>/idp/sts.wst?
TokenProcessorId=saml2firstinstance
```

Additional Token Processors may be *downloaded* from the Ping Identity web site (`www.pingidentity.com/en/products/downloads.html`). For configuration information, please consult documentation provided for the respective add-on processor.

To begin configuring Token Processors:

1. Click **IdP Configuration** on the Main Menu.
2. Click **Token Processors** under Application Integration Settings.

   If this link is not shown, ensure that the WS-Trust STS is enabled in **Server Settings** (see *Enable the WS-Trust STS* on page 378).

To configure a new token-processor instance:

- Click **Create New Instance**.

To edit an existing instance:

- Click the Instance Name and click the step you need to change.

To delete an instance:

1. Click **Delete** next to the Instance Name. (To undo the deletion, click **Undelete**.)

   Note: This option is available only if the processor instance is not in use for a connection.

2. Click **Save** to confirm the deletion.

Select a token processor type

The first step in creating a token-processor instance is choosing the processor type.

To define an instance:

1. Enter the Instance Name and Instance Id on the Type screen.
2. Select the processor Type from the drop-down menu.
3. Optional: Select a **Parent Instance** from the list.

   If you are creating an instance that is similar to an existing instance, you might consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.

4. Click **Next** and enter information on the configuration screen for this token-processor instance.

   This configuration varies depending on the token processors deployed on your server. For add-on processors please consult the online documentation referenced in the download package, or look under *Product Documentation* at pingidentity.com.

   For token processors bundled with PingFederate, refer to one of the following sections:

   - *Configure a Username Token Processor instance* on page 383
   - *Configure a Kerberos Token Processor instance* on page 383

- *Configure an OAuth Token Processor instance* on page 383
- *Configure a JSON Web Token (JWT) Processor instance* on page 384
- *Configure a SAML Token Processor instance* on page 384

Configure a token processor instance

This configuration varies depending on the token processors deployed on your server. For add-on processors please consult the online documentation referenced in the download package, or look under *Product Documentation* at pingidentity.com.

For token processors bundled with PingFederate, refer to one of the following sections:

- *Configure a Username Token Processor instance* on page 383
- *Configure a Kerberos Token Processor instance* on page 383
- *Configure an OAuth Token Processor instance* on page 383
- *Configure a JSON Web Token (JWT) Processor instance* on page 384
- *Configure a SAML Token Processor instance* on page 384

Configure a Username Token Processor instance

The integrated Username Token Processor accepts and validates username security tokens.

Note: If this is a child instance, select the override check box to modify the configuration.

To configure the token-processor instance for Username Token validation:

1. On the Instance Configuration screen, click **Add a new row to 'Password Credential Validators'** to define a credential-authentication mechanism instance for the adapter.
2. Select a password credential validator from the list and click **Update**.

   Add as many validators as necessary. Use 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 defined password credential validators 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 validators, the failovers could lockout those accounts in their source locations.
3. Click **Next**.

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 (AD) domain service, thus making it possible to restrict access to users authenticating through specific mechanisms. For more information, see *Authentication mechanism assurance* on page 425.

Note: If this is a child instance, select the override check box to modify the configuration.

1. 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 179 .

   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 179).
2. Click **Next**.

Configure an OAuth Token Processor instance

The PingFederate STS provides validation for OAuth Bearer tokens (see *PingFederate OAuth AS* on page 60).

Generally, a client would send a base64-encoded access token in order to receive a SAML token in exchange.

Note: To use this token processor, you must first configure an access-token attribute contract (see *OAuth access token management* on page 186).

Note: If this is a child instance, select the override check box related to the settings you want to modify.

Field descriptions

Field	Description
Access Token Manager	Select the Access Token Management plug-in instance that will be used to validate the OAuth Bearer Access Token.
Scope Value as Single String (optional)	If selected, the Scope value will be made available as a single space delimited set of string values. Alternatively, Scope values will be made available as a multivalue attribute.

To configure the token-processor instance for OAuth Bearer token validation:

1. Select an Access Token Manager.
2. Optional: Select the Scope Value as Single String check box.
3. Click **Next**.

Configure a JSON Web Token (JWT) Processor instance

The PingFederate STS provides validation for JSON web tokens.

Note: If this is a child instance, select the override check box to modify the configuration.

Field descriptions

Field	Description
JWKS Endpoint URI	The URI of the JWKS endpoint. A set of JSON Web Keys (JWK) are downloaded from this endpoint and used for JWT signature verification.
Issuer	(Required) 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.

To configure the token-processor instance for JWT validation:

1. Enter a JWKS Endpoint URI.
2. Enter the Issuer.
3. Enter an Expiry Tolerance.
4. Click **Next**.

Configure a SAML Token Processor instance

On the Instance Configuration screen, you may use signing-certificate DN checking to limit the valid signatures and certificates for token requests accepted for this SAML token type. (By default, the STS validates digital signatures using all trusted Certificate Authorities (CAs) imported into PingFederate.)

At minimum on this screen, you must indicate a unique identifier for the PingFederate STS. To be accepted, an incoming SAML token must contain this ID in its <audience> element.

Note: If this is a child instance, select the override check box to modify the configuration.

To configure the token-processor instance for certificate validation:

1. Enter a URI for Audience.

   This is the ID for the STS for either SAML 1.1 or SAML 2.0 tokens, depending on which processor you are configuring (see *Specify federation information* on page 137).
2. Optional: Click the **Add a new . . .** link under Action for either Valid Certificate Issuer DNs or Valid Certificate Subject DNs.

   You can use both lists.

   Important: When both types of validation are configured, then the certificate used to validate signatures must match an entry in *both* lists. If only Subject DNs are listed on this screen, then the certificate Issuer DN is not checked and its Subject DN must match one of the entries in the Subject DNs list. If only Issuer DNs are listed here, then the certificate Subject DN is not checked and its Issuer DN must match one of the entries in the Issuer DNs list. If neither Issuer DNs nor Subject DNs are listed, then all certificates are treated as valid for purposes of verification.
3. Optional: Enter a Valid DN and click **Update**.
4. Optional: Repeat the previous steps as needed to add more DNs.

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—an extended processor-attribute contract.

Note: This screen shows a different list of attributes for the Core Contract, depending on the Token Processor selected.

To add an attribute:

Note: If this is a child instance, select the override check box to modify the configuration.

- Optional: Enter the attribute name in the text box and click **Add**.

  Important: For the OAuth 2.0 Bearer Token Processor, added attributes must also be among those configured under Access Token Management (see *Define the access token attribute contract* on page 190).

Set attribute masking

On the Token Attributes screen, you can choose to mask attribute values that PingFederate logs from this processor instance at runtime (see *Attribute masking* on page 71).

To mask an attribute in log files:

Note: If this is a child instance, select the override check box to modify the configuration.

- Optional: Under Mask Log Values select the attribute whose value you want to mask.

  If OGNL expressions might be used to map derived values into outgoing tokens and you want those values masked, select the related check box under the Attribute list (see *Attribute mapping expressions* on page 485).

Review the token processor configuration

From the Summary screen, you can reach processor 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 Token Processors screen.

To save a processor instance:

1. Click **Done** on the Summary screen.
2. Click **Save** on the Manage Token Processors screen.

Manage STS request parameters

As an option for configuring PingFederate to act as a WS-Trust STS, an administrator 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 for SP-partner connections (see *Select a request contract* on page 389).

- To reach this screen, click **STS Request Parameters** under Application Integration Settings on the IdP Configuration menu.

  If this link is not present, WS-Trust is not enabled (see *Enable the WS-Trust STS* on page 378).
- To add a new set of request parameters, click **Add New Request Contract**.
- To edit an existing contract, click its **Contract Name**.

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—see *Manage STS request parameters* on page 386).

Field descriptions

Field	Description
Contract Name	A descriptive name for the Contract—for example, a Web Service Client or Provider.
Contract ID	An internal identifier—must be alphanumeric with no spaces.
Parameters to be provided in the request	A list of request parameters for this Contract (see instructions below).

To add a Parameter:

- Enter the Parameter Name in the text box and click **Add**.

To modify a Parameter Name:

1. Click **Edit** under Action for the Parameter Name.
2. Edit the name and click **Update**.

   Note: If you change your mind, be sure to click the **Cancel** *link* in the Actions column, not the **Cancel** *button*, which discards any other changes you might have made.

To delete a Parameter:

- Click **Delete** for the Parameter Name.

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 enable STS for a new connection, or to add the capability to an existing connection:

1. Select the WS-Trust STS option on the Connection Type screen (see *Choose an SP connection type* on page 250).

   Note: Before this option can be selected, the WS-Trust protocol must be enabled in Server Settings (see *Server settings* on page 378).
2. Select a Default Token Type.

   The Default Token Type, either SAML 1.1 or 2.0, is used when a Web Service client does not specify in the token request what token type the STS should issue.

Note: The Default Token Type *does not* need to match the Protocol indicated on the screen for SSO (when applicable).

When the option is enabled, the configuration starts on the WS-Trust STS screen.

- To continue, click **Configure WS-Trust STS**.

Configure protocol settings for IdP STS

On this screen, use the **Add** button for the Partner Service Identifier field to enter one or more URLs for your partner's Web Service(s). Each of these identifiers is compared to the element `<AppliesTo>` in Requests for Security Tokens (RSTs) and may be either a complete URL or a base URL for matching variable ports or paths.

Also on this screen, options are available for adding signature or encryption protection to outgoing SAML tokens and for enabling support of two additional token-type requests:

- For SAML 1.1 and SAML 2.0, you can choose to generate a symmetric key to be used in conjunction with the "Holder of Key" designation for the assertion's Subject Confirmation Method (for information about HoK assertions, see, for example, *Web Services Security SAML Token Profile* (docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0.pdf).
- For SAML 2.0, you can choose to encrypt the assertion.
- Enabling the OAuth SAML Bearer Profile permits two additional token-type requests based on these OAuth grant types:
  - SAML 2.0 Bearer Assertion Grant Type
  - OAuth Access Token via SAML 2.0 Bearer Assertion Grant Type

  See *STS OAuth integration* on page 59 for more information on the use of these token-type requests.

  Note: You can make any or all selections if you expect requests for these types of tokens. These selections are independent of the Default Token Type selected previously (see *Configure SP connections for STS* on page 386).

When you make one (or all) of these selections, you are asked to choose a signing or XML encryption certificate later in the connection setup, unless required certificates are already in place for an existing browser-based SSO connection. (PingFederate uses the same certificates to handle signing/encryption requirements for both Browser SSO and WS-Trust STS—for more information, see *Configure credentials* on page 289.)

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.

Field Descriptions

Field	Description
Minutes Before	The amount of time before the token was issued during which it is to be considered valid.
Minutes After	The amount of time after the 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**.

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 (see *Fulfill the attribute contract for token*

creation on page 399). 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 68). You identify these attributes on this screen.

(This screen presents an additional option when SAML 1.1 is chosen as the Default Token Type on the Connection Type screen.)

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **Attribute Contract on the Summary screen**.

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 (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 399).

   For information about the NameID elements and applicable URI values, locate the SAML 2.0 specification at *oasis-open.org/specs*.

   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 399).
2. Optional: For SAML 1.1 tokens, select the Attribute Namespace.

   This field appears only when SAML 1.1 is chosen as the Default Token Type on the Connection Type screen (see *Configure SP connections for STS* on page 386).

   Change the default Namespace selection if you and your SP partner have agreed to a specific namespace (see *STS namespaces* on page 69).

   Note: If needed, an administrator can customize namespace alternatives via the `custom-name-formats.xml` configuration file located in this directory:

   `<pf_install>/pingfederate/server/default/data/config-store`
3. Click **Add**.

To modify an attribute name or namespace:

1. Click **Edit** under Action for the attribute.
2. Make the change and click **Update**.

To delete an attribute:

- Click **Delete** under Action for the attribute.

Select a request contract

This optional setting allows you to use XML parameters contained in RSTs for token-attribute mapping (see *Manage STS request parameters* on page 386).

- 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 the drop-down list.

  If the contract you want is not shown, click **Manage STS Request Parameters**.

  When you choose a contract, you will enable an option to select Request from the drop-down Source list on the attribute-mapping screen (see *Fulfill the attribute contract for token creation* on page 399).

Manage IdP token processor mappings

IdP token processors are responsible for validating incoming security tokens as part of an STS operation (see *Token processors and generators* on page 58). A configured and deployed token processor in PingFederate is known as a token processor instance. The same instance may be mapped by multiple connections.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.

To modify an existing Token Processor Instance:

- Click its Name link.

To begin configuring a Token Processor Instance for this connection:

- Click **Map New Token Processor Instance**.

Select a token processor instance

On this screen for a new connection, choose an instance of the Token Processor needed for this connection (see *Token processors and generators* on page 58).

You will use attributes returned from the token processor (the token-processor contract) to fulfill the attribute contract required for this partner and/or use them to look up additional attributes in a user-data store. You make this choice on the next screen (see *Select an attribute retrieval method for token generation* on page 408).

- Choose a Token Processor Instance from the drop-down list and click **Next** to continue.

  (Optional) To override any token generator instance, select the **Override Instance Settings** check box (see *Overriding Token Generator Instances*).

  To create or change a processor instance, as needed, click **Manage Token Processor Instances**.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.

Override a token processor instance

Overrides at the connection level simplify token management by allowing you to create a small set of token instances that define the base configuration requirements and then overriding settings at the connection level as needed.

You may override any token processor settings at the connection level either during or after connection mapping.

Alternatively, you can override any token processor instance to apply to several connections, see *Hierarchical plug-in configurations* on page 66 for more information about token processor overrides.

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, make your changes, and then click **Next**. When you are finished, click **Done** to continue with token mapping.

  Note: Override token-setting screens are functionally identical to those used for creating a new token connection. Refer to the table below to find sections in this manual containing configuration information and procedures.

  The display of some screens listed in the table depends on the type of token you are configuring.

  For information about the override token-setting screens, depending on the type of setting, use the following table:

Override Screen	Manual Section
Instance Configuration	See *Manage token processors* on page 381.
Extended Contract	See *Extend a token processor contract* on page 385.
Token Attributes	See *Set attribute masking* on page 385.

- To remove any token processor connection override, clear the **Override Instance Settings** check box, and then complete the rest of the setup.

Review the token processor type

Token processor override instance type information. This screen is view only.

This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Configure a token processor instance* on page 383.

Override token processor settings

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Configure a token processor instance* on page 383.

Override the token processor extended contract

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Extend a token processor contract* on page 385.

Override the token processor attributes

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Set attribute masking* on page 385.

Review the overridden token processor settings

- On the Override Instance Settings Summary screen, click any heading to change your processor configuration; or click **Done** to continue.

Restrict a token processor to certain virtual server IDs

When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), you have the option to enforce authentication requirements by restricting a token processor to certain virtual server IDs. This optional setting can be applied to each token processor added to the connection using virtual server IDs. By default, no restriction is imposed.

To restrict a token processor to a subset of the available virtual server IDs:

1. Select the **Restrict Virtual Server IDs** check box.
2. In the **Allowed Virtual Server IDs** area, select virtual server IDs that you want to allow for this token processor.
3. Click **Next**.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Virtual Server IDs** on the Summary screen.

   Note: The Virtual Server IDs screen is only available for connections using at least one virtual server ID, see *Federation Server Identification*.

Select an attribute retrieval method for token creation

For token creation, you can query local user-data stores to help fulfill the attribute contract, in conjunction with attribute values supplied by the token processor you are using with PingFederate (see *Manage token processors* on page 381).

The values supplied by the token processor are shown under Token Processor Contract on the Attribute Retrieval screen.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Attribute Retrieval** on the Summary screen.

- If you choose to look up additional information, then you will identify a data store and specify lookup queries next (see the next section *Configure a data store for token generation* on page 408).
- If you use only the attributes available (the default), then you will map values for the attribute contract next (see *Fulfill the attribute contract for token generation* on page 413).

Tip: To determine whether you need to look up additional values, compare the attribute contract against the token-generator contract on the previous screen (see *Select a token generator instance* on page 406). If the token-generator contract requires more information, determine whether your local data stores can supply it. (You can also choose to use text constants or expressions for certain information—see *Fulfill the attribute contract for token generation* on page 413.)

Configure attribute sources and user lookup for token creation

Attribute sources are specific database, directory, or custom data store locations containing information that may be needed for the attribute contract (see *Define an attribute contract for IdP STS* on page 388). Attribute sources can be reused across connections to other SP partners.

This portion of the connection configuration allows you to configure one or more data stores to look up attributes and to set up search parameters.

Note: Queries are executed in the order of Attribute Sources shown. Use the move up/move down controls as needed to adjust the order.

To configure an attribute source:

- Click **Add Attribute Source** and complete the setup steps (see *Configure a data store for token creation* on page 393).

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 **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.

5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **Attribute Source & User Lookup**.

   If this step is not listed, then this instance is configured to use token-processor values only (see *Select an attribute retrieval method for token creation* on page 391).

Configure a data store for token creation

This screen allows you to choose a data store from a previously configured list (see *Manage data stores* on page 143). Attribute values extracted from one or more data stores are used to help fulfill the attribute contract (see *Define an attribute contract for IdP STS* on page 388).

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **Attribute Source & User Lookup** under the IdP Token Processor Mapping tab.

   If this step is not listed, then this instance is configured to use token-processor values only (see *Select an attribute retrieval method for token creation* on page 391).
10. Click the attribute source Description link.

To define an attribute source:

1. Enter an Attribute Source Id to uniquely identify the data source for the mapping.
2. Use Attribute Source Description to specify an attribute source name that distinguishes this user lookup for the selected data store.

   Note: PingFederate appends this description to the data store type in the Source list on the Attribute Contract Fulfillment screen (see *Fulfill the attribute contract for token creation* on page 399).
3. Choose an Active Data Store and click **Next**.

   A data-store configuration must be defined under System Settings for use within a connection. If the data store you want is not shown in the drop-down menu, click **Manage Data Stores** to add it (see *Manage data stores* on page 143).

Set up an attribute source for token creation

See the following sections in this manual, depending on the type of data store:

Data Store Type	Related Manual Section
JDBC	• *Select a database table and columns for token creation* on page 394 • *Specify a database filter for token creation* on page 395
LDAP	• *Configure an LDAP search for token creation* on page 396 • *Specify an LDAP filter for token creation* on page 397

Data Store Type	Related Manual Section
Custom	• *Configure custom source filters for token creation* on page 398 • *Select custom source fields for token creation* on page 398

Select a database table and columns for token creation

When you choose to use a database source for attributes, you follow this path through the configuration steps.

On this screen you begin to specify exactly where additional data can be found to complete the attribute contract when you send a security token to this SP (see *Define an attribute contract for IdP STS* on page 388). Only one table may be used as a source of data for a JDBC lookup.

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 *http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html*.

Field descriptions

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 a JDBC query. Other databases, such as MySQL, do not require selection of a schema.
Table	The name of the table contained in the database. Use the drop-down to change the table.
Columns to return from SELECT	Displays the columns available from the selected table. Select the columns that are associated with the desired attributes you would like to return from the JDBC query.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **Database Table and Columns** under the IdP Token Processor Mapping tab.

To select a database table and columns for queries:

1. Choose a Schema file (when applicable) from the drop-down list.
2. Choose a Table from the drop-down list.
3. Choose a name under Columns to Return from Select and click **Add Attribute**.

Tip: Click **Refresh** if you are updating an existing configuration and changes may have been made to the database.

Repeat this step for other columns as needed.

Note: You do not need to add a column here for it to be used as part of a search filter (see *Specify a database filter for token generation* on page 410 next). Add only attributes from which you need actual values to pass in a token.

Tip: To determine what attributes to look up during a query, click the **View Attribute Contract** link to see what information must be collected (see *Define an attribute contract for IdP STS* on page 388). Then determine what information is coming in from the token processor (see *Select an attribute retrieval method for token creation* on page 391). Information not contained in the token-processor contract may be pulled from the data store look-up query.

Specify a database filter for token creation

The JDBC `WHERE` clause in PingFederate queries the data table you selected to retrieve a record associated with a particular value (or values) from the incoming security token. The clause is in the form:

`WHERE column1=value1 [AND column2=value2] [OR ...]`

The left side of the first variable pair uses a column name in the database table you selected (see *Select a database table and columns for token creation* on page 394).

The right side generally uses values passed in from a token processor (variables, including the correct formatting, are listed under Token Processor Values—see *Manage token processors* on page 381).

Note: 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. For more information on multiple data-store mapping, see *Multiple data source attribute mapping* on page 70.

You can also apply additional search criteria from your own database, using any other columns from the targeted table.

Tip: Click "**View List of Columns . . .**" to see a list from which to copy and paste.

For more information about `WHERE` clauses, consult your DBMS documentation.

EXAMPLE:

```
userid='${username}'
```

In this example `userid` is the name of a column in the JDBC data store. On the right side, `'${username}'` returns the value of the `username` variable from the IdP token processor.

Important: You *must* use the `${}` syntax to retrieve the value of the enclosed variable and use single quotation marks around the `${}` characters.

Field descriptions

Field	Description
Where	`WHERE` clause statements conditionally select data from a table. Enter the `WHERE` clause statement in the space provided. For example: `WHERE email='clive@company.com'`.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.

3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **Database Filter** under the IdP Token Processor Mapping tab.

To construct the `WHERE` clause:

1. Enter the statement in the space provided, following the guidelines and example above.

   The initial `WHERE` is optional.
2. Ensure the syntax and variable names are correct.

   When you click **Next**, you will map attribute values returned from the database into the security token (see *Fulfill the attribute contract for token creation* on page 399).

Configure an LDAP search for token creation

When you choose to use an LDAP source for attributes, you follow this path through the configuration steps.

On this screen you specify the branch of your LDAP hierarchy where you want PingFederate to look up user data.

Field descriptions

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	Determines the node depth of the query. Select Subtree, One level or Object.
Root Object Class	The class containing the attributes you want.
Attributes to return from search	A list of attributes added from the drop-down list below. Subject DN is a default attribute, which may be used as the primary user identifier.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **LDAP Directory Search** under the IdP Token Processor Mapping tab.

To select LDAP attributes:

1. Optional: Enter a Base DN.
2. Select a Search Scope.
3. Select a Root Object Class.
4. Under Attributes to return from search, choose an attribute and click **Add Attribute**.

Note that the attribute Subject DN is always returned by default.

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 as well as indirectly through nested group membership (if any) under the Base DN.

For example, suppose you have three groups under the Base DN, namely 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).

5. Repeat the last step for other attributes as needed.

Note: You do not need to add an attribute here for it to be used in a search filter (see *Specify an LDAP filter for token creation* on page 397). Add only attributes from which you need actual values to pass into the outgoing security token.

Specify encoding for LDAP binary attributes for token creation

The LDAP Binary Attribute Encoding Types screen appears when a binary attribute is added on the LDAP Directory Search screen. Because binary attribute data cannot be used in an assertion, use this screen to specify which encoding type (Base64, Hex or SID) you want to apply during attribute contract fulfillment.

For example, Microsoft Office 365 uses objectGUID, an immutable Active Directory binary attribute associated with user accounts. Office 365 requires this binary data to be Base64-encoded to correlate provisioned federated user data to Active Directory accounts.

Claims-based authentication with Microsoft Outlook Web App and Exchange admin center (EAC) is another example. It requires tokenGroups (another binary attribute in Active Directory) to be SID-encoded.

Note: Attributes are specified as binary when configuring an LDAP connection (see *Specify LDAP binary attributes* on page 148).

To select an attribute encoding type:

- Select the type of attribute encoding you want to apply for each attribute and click **Next**.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **LDAP Binary Attribute Encoding Types** under the IdP Token Processor Mapping tab.

Specify an LDAP filter for token creation

The LDAP filter queries the data you selected to retrieve a record associated with a particular value (or values) from the incoming token. The filter is in the form:

```
attribute=${value}
```

The left-side variable is an attribute you selected earlier (see *Configure an LDAP search for token creation* on page 396). The right side generally uses values passed in from the security token (variables, including the correct syntax, are listed under Security Token Values—see *Manage token processors* on page 381).

You can also apply additional search criteria from your data store, using any other attributes from the targeted object classes.

Tip: Click "**View List of Available LDAP Attributes**" for a list from which you can copy and paste.

For general information about search filters, consult your LDAP documentation.

Field descriptions

Field	Description
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: 1) at least one attribute (attribute data type) to search on, 2) a search filter operator that will determine what to match, and 3) the value of the attribute being sought. Searches must have at least one of each of these three components.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **LDAP Filter** under the IdP Token Processor Mapping tab.

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.

   Note: If you used an anonymous binding to create this LDAP connection, your access might be restricted (see *Configure an LDAP connection* on page 145).
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

Configure custom source filters for token creation

When you choose to use a custom source for attributes, you follow this path through the configuration steps.

On this screen you specify a filter, or lookup 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.

Select custom source fields for token creation

On the Configure Custom Source Fields screen, you can choose from among the fields shown to map to the attribute contract. These choices are supplied by the driver implementation. Select only those needed to fulfill the attribute contract for this partner connection.

Review data lookup configuration for token creation

Use this screen to view and edit the configuration as needed. Click any heading to change the settings.

Fulfill the attribute contract for token creation

You map attributes for outgoing security tokens for this partner on the Attribute Contract Fulfillment screen.

Map each attribute to fulfill the Attribute Contract from one of these Sources:

- Token

  When you make this selection, the associated Value drop-down list is populated by the token processor.
- LDAP/JDBC/Custom

  Note: PingFederate appends a description in parentheses for configured data store lookups (see *Configure attribute sources and user lookup for token creation* on page 392).

  Values are returned from your attribute source (if you are using data store—see *Select an attribute retrieval method for token creation* on page 391). When you make this selection, the Value list is populated by the LDAP, JDBC, or Custom attributes you identified as an Attribute Source (see *Configure an LDAP search for token creation* on page 396, *Select a database table and columns for token creation* on page 394, or *Configure custom source filters for token creation* on page 398).
- Context

  Values are returned from the context of the transaction at runtime.

  Note: The HTTP Request and STS SSL Client Certificate Chain selections are retrieved as 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 (see *Using the OGNL edit screen* on page 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)

  Note: When using the STS Basic Authentication Username, STS SSL Client Certificate's Subject DN, or STS SSL Client Certificate Chain attributes, ensure the associated authentication is enabled and configured in WS-Trust STS Settings (see *Select authentication methods* on page 379).
- Request

  Values are supplied from parameters in the token request received from the Web Service Client. This selection is available only if a Request Contract was selected earlier (see *Select a request contract* on page 389).
- 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 (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 incoming token assertion, 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 *Configure attribute sources and user lookup for token creation* on page 392) and attribute is any of the data store attributes you select.

To reach this screen for editing:

1. Click **IdP Configuration** on the Main Menu.
2. Click the connection name under SP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the SP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Creation** under the WS-Trust STS tab.
6. Click **Configure Token Creation**.
7. Click **IdP Token Processor Mapping** on the Summary screen.
8. Click the Token Processor Instance Name.
9. Click **Attribute Contract Fulfillment** under the IdP Token Processor Mapping tab.

To map attributes:

1. Choose a Source for each Target attribute.
2. Choose (or enter) a Value for each Attribute.

   See *Map each attribute to fulfill the Attribute Contract from one of these Sources:* above. All values must be mapped.
3. Click **Next**.

Specify issuance criteria for token creation (optional)

Use this screen to define criteria PingFederate can evaluate to determine whether to issue a SAML security token for a user (see *About token authorization* on page 71). This token authorization can be used to restrict who can access identity-enabled Web Services.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values returned from the context of the transaction at runtime.

     Note: The HTTP Request and STS SSL Client Certificate Chain selections are retrieved as Java objects rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values. If you want to use an expression for authentication, skip to Step 8 below.
   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for configured data store lookups (see *Configure attribute sources and user lookup* on page 265)
   - Mapped Attributes – Select to access the required attributes.
   - Request – Select to access parameters from a request contract.
   - Token – Select to access the required attributes from the attribute contract.
2. Select an attribute name.

   Note: When using the STS Basic Authentication Username, STS SSL Client Certificate's Subject DN, or STS SSL Client Certificate Chain attributes, ensure the associated authentication is enabled and configured in WS-Trust STS Settings (see *Select authentication methods* on page 379).
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

   Important: When using the STS SSL Client Certificate's Subject DN attribute, you must select one of the following conditions: equal to DN, not equal to DN, multi-value contains DN, or multi-value does not

contain DN. These operators normalize the DN before comparison to accommodate for different string representations that are still considered equivalent (for example, case sensitivity, or whitespace).

4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).

5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   - Errors result in a `SOAP` message returned. 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`).

     Note: 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.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Enable and disable expressions* on page 485).

   Important: When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), consider using an OGNL expression to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access (see *Sample OGNL expressions* on page 487).

   - Use the in-line editor box to enter the OGNL expression.

     For more information about OGNL, see *Attribute mapping expressions* on page 485.
   - Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.

   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

     Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

Review the IdP token processor mapping

When you have finished configuring IdP Token Processor 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.

- If you are editing an existing connection, click **Done** on successive screens until you reach the WS-Trust STS screen, and then click **Save**.

To save a new configuration:

1. Click **Done** to return to the IdP Token Processor Mapping screen.
2. Click **Next** to go to the Token Creation Summary screen, and then click **Done**.
3. On the Token Creation screen, click **Done**.
4. On the WS-Trust STS screen, 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, you can choose whether to continue or abort the token-creation process.

Note: The Error Handling screen is presented only if a Request Contract is used for this configuration (see *Select a request contract* on page 389).

Review the token creation configuration

When you have finished configuring Token Creation, 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.

- If you are editing an existing connection, click **Done** on successive screens until you reach the WS-Trust STS screen, and then click **Save**.

Review the IdP STS configuration

On the WS-Trust STS Summary screen, you can review the configuration for this connection.

- If you need to make any changes to a new or existing connection, click the heading over the information you want to modify.

SP configuration for STS

This section covers the SP configuration for STS, including:.

- *Manage token generators* on page 402
- *Configure IdP connections for STS* on page 404

Manage token generators

Token Generators are used to issue security tokens that can be consumed by Web Services at your site (see *Token processors and generators* on page 58). Token Generators for SAML 2.0 and SAML 1.1 tokens are included with the PingFederate installation. This section provides guidance on configuring “instances” of either of the SAML Token Generators. You must configure at least one generator in order to set up an STS connection or token-to-token mapping (see *Token translator mappings* on page 421).

Important: If more than one instance of the same token-generator type is configured for a connection or token-to-token mapping, clients calling either of the PingFederate STS endpoints must add a query parameter, `TokenGeneratorId`, and specify the Instance Id. (For endpoint information, see *View SP protocol endpoints* on page 317.)

For example: `https://<pf_host>:<pf_port>/sp/sts.wst?TokenGeneratorId=saml2firstinstance`

Additional Token Generators may be *downloaded* from the Ping Identity web site (`www.pingidentity.com/en/products/downloads.html`). For configuration information, please consult documentation provided for the respective add-on generator.

To begin configuring SAML 1.1 or 2.0 Token Generators:

1. Click **SP Configuration** on the Main Menu.
2. Click **Token Generators** under Application Integration Settings.

If this link is not shown, ensure that the WS-Trust STS is enabled in **Server Settings** (see *Enable the WS-Trust STS* on page 378).

To configure a new token-generator instance:

- Click **Create New Instance**.

To edit an existing instance:

- Click the Instance Name and click the step you need to change.

To delete an instance:

1. Click **Delete** next to the Instance Name. (To undo the deletion, click **Undelete**.)

   Note: This option is available only if the generator instance is not in use for a connection.

2. Click **Save** to confirm the deletion.

Select a token generator type

The first step in creating a SAML token-generator instance is choosing the generator type.

To define an instance

1. Enter the Instance Name and Instance Id on the Type screen.
2. Select SAML 1.1 Token Generator or SAML 2.0 Token Generator from the drop-down menu.
3. Optional: Select a **Parent Instance** from the list.

   If you are creating an instance that is similar to an existing instance, you might consider making it a child instance by specifying a parent. A child instance inherits the configuration of its parent unless overridden. You can specify overrides during the rest of the setup.
4. Click **Next**.

Configure a token generator instance

On the Instance Configuration screen, you specify parameters for generated SAML tokens.

Note: If this is a child instance, select the override check box to modify the configuration.

Field Instructions

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 the SAML 2.0 Entity ID or SAML 1.x Issuer specified on the Federation Information screen in Server Settings (see *Specify federation information* on page 137).
Signing Certificate	Responses containing SAML tokens must be signed. If the signing certificate you need is not in the drop-down list, click **Manage Signing Certificates** near the bottom of the screen.
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.

Field	Instructions
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	(Optional) Choose from among available methods: • ...cm:sender-vouches (default) • ...cm:bearer • ...cm:holder-of-key For more information, see the *WSS SAML Token Profile*.
Encryption Certificate	The WSP's public certificate for encryption, required *only* if holder-of-key is selected as the Confirmation Method. If the certificate is not yet part of the PingFederate store, click **Manage Encryption Certificates** to import it.

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.

Note: If this is a child instance, select the override check box to modify the configuration.

To add an attribute:

- Enter the attribute name in the text box and click **Add**

Review the token generator configuration

From the Summary screen, you can reach token-generator 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 Token Generators screen.

To save a generator instance:

1. Click **Done** on the Summary screen.
2. Click **Save** on the Manage Token Generators screen.

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 enable STS for a new connection, or to add the capability to an existing connection:

- Select the WS-Trust STS option on the Connection Type screen (see *Choose an SP connection type* on page 250).

  Note: Before this option can be selected, the WS-Trust protocol must be enabled in Server Settings (see *Server settings* on page 378).

When the option is enabled, the configuration starts on the WS-Trust STS screen.

- To continue, click **Configure WS-Trust STS**.

Configure protocol settings for SP STS

On the Protocol Settings screen, choose whether to validate incoming SAML tokens or to validate and then also generate different tokens to enable SSO access to Web Services at your site.

- If you choose not to generate new tokens, then no further settings are needed for this task—click **Next** and refer to *Review the token generation configuration* on page 416 for instructions on saving this configuration.You will be asked later to choose a certificate with which to verify the signature on the incoming SAML token (see *Configure signature verification settings* on page 292).

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 (the attribute contract). The attribute values from the incoming token can be then mapped to attributes in the token generator you select (see *Configure attribute contract fulfillment* on page 273). 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 (see *Attribute contracts* on page 68). You identify these attributes on this screen.

Optionally, you can mask the values of attributes (other than `SAML_SUBJECT`) in the log files that PingFederate writes when it receives security tokens (see *Attribute masking* on page 71).

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Attribute Contract** on the Summary screen.

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 the requester.
2. Optional: Select the check box under Mask Values in Log.
3. Click **Add**.

To modify an attribute name :

1. Click **Edit** under Action for the attribute.
2. Make the change and click **Update**.

To delete an attribute:

- Click **Delete** under Action for the attribute.

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.

You can map one or more token generator instances into each IdP connection to satisfy multiple session-management requirements where needed. The same instances may be mapped by multiple connections.

When token generators are restricted to certain virtual server IDs, the allowed IDs are displayed under the Virtual Server IDs column.

The configuration begins on the Token Generator Mapping & User Lookup screen. If you have not yet configured an instance of a token generator you need for this connection, see *Manage token generators* on page 402.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.

To modify an existing Token Generator Instance:

- Click its Name link.

To begin configuring a Token Generator Instance for this connection:

- Click **Map New Token Generator Instance**.

Select a token generator instance

On this screen for a new connection, choose an instance of the Token Generator needed for this connection (see *Token processors and generators* on page 58).

You will use attributes contained in the incoming security token to fulfill the token generator contract for this STS connection and/or use them to look up additional attributes in a user-data store. You make this choice on the next screen (see *Select an attribute retrieval method for token creation* on page 391).

- Choose a Token Generator Instance from the drop-down list and click **Next** to continue.

  (Optional) To override any token generator instance, select the **Override Instance Settings** check box (see *Overriding Token Generator Instances*).

  To create or change a Token Generator Instance, as needed, click **Manage Token Generator Instances**.

Override a token generator instance

Overrides at the connection level simplify token management by allowing you to create a small set of token instances that define the base configuration requirements and then overriding settings at the connection level as needed.

You may override any IdP token generator settings at the connection level either during or after connection mapping.

Alternatively, you can override any token generator instance to apply to several connections, see *Hierarchical plug-in configurations* on page 66 for more information about token generator overrides.

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, make your changes, and then click **Next**. When you are finished, click **Done** to continue with token mapping.

  Note: Override token-setting screens are functionally identical to those used for creating a new token connection. Refer to the table below to find sections in this manual containing configuration information and procedures.

  The display of some screens listed in the table depends on the type of token you are configuring.

  For information about the override token-setting screens, depending on the type of setting, use the following table:

Override Screen	Manual Section
Instance Configuration	See *Configure a token generator instance* on page 403
Extended Contract	See *Extend a token generator contract* on page 404.

- To remove any token generator connection override, clear the **Override Instance Settings** check box, and then complete the rest of the setup.

Review the token generator type

Token generator override instance type information. This screen is view only.

This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Configure a token generator instance* on page 403.

Override token generator settings

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Configure a token generator instance* on page 403.

Override the token generator extended contract

- If you want to override any settings on this screen, select the override check box, make your changes, and then click **Next**.

  This screen is functionally identical to the one used for creating a new token instance. For more information about this screen, see *Extend a token generator contract* on page 404.

Review the overridden token generator settings

- On the Override Instance Settings Summary screen, click any heading to change your generator configuration; or click **Done** to continue.

Restrict a token generator to certain virtual server IDs

When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), you have the option to enforce authentication requirements by restricting a token generator to certain virtual server IDs. This optional setting can be applied to each token generator added to the connection using virtual server IDs. By default, no restriction is imposed.

To restrict a token generator to a subset of the available virtual server IDs:

1. Select the **Restrict Virtual Server IDs** check box.
2. In the **Allowed Virtual Server IDs** area, select virtual server IDs that you want to allow for this token generator.
3. Click **Next**.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Virtual Server IDs** on the Summary screen.

   Note: The Virtual Server IDs screen is only available for connections using at least one virtual server ID, see *Federation Server Identification*.

Select an attribute retrieval method for token generation

For token generation, you can query local user-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.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Attribute Retrieval** on the Summary screen.

- If you choose to look up additional information, then you will identify a data store and specify lookup queries next (see the next section *Configure a data store for token generation* on page 408).
- If you use only the attributes available (the default), then you will map values for the attribute contract next (see *Fulfill the attribute contract for token generation* on page 413).

Tip: To determine whether you need to look up additional values, compare the attribute contract against the token-generator contract on the previous screen (see *Select a token generator instance* on page 406). If the token-generator contract requires more information, determine whether your local data stores can supply it. (You can also choose to use text constants or expressions for certain information—see *Fulfill the attribute contract for token generation* on page 413.)

Configure a data store for token generation

This portion of the connection configuration allows you to set up search parameters for a data store.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Data Store** under the Token Generator Mapping & User Lookup tab.

   If this step is not presented, this Token Generator Instance is not configured to look up user attributes in a data store (see *Select an attribute retrieval method for token generation* on page 408).

To define an attribute source:

- Choose an Active Data Store and click **Next**.

  A data-store configuration must be defined under System Settings for use within a connection. If the data store you want is not shown in the drop-down menu, click **Manage Data Stores** to add it (see *Manage data stores* on page 143).

Set up an attribute source for token generation

See the following sections in this manual topics, depending on the type of data store:

Data Store Type	Related Manual Section
JDBC	• *Select a database table and columns for token generation* on page 409 • *Specify a database filter for token generation* on page 410
LDAP	• *Configure an LDAP search for token generation* on page 411 • *Specify an LDAP filter for token generation* on page 412
Custom	• *Configure custom source filters for token generation* on page 413 • *Select custom source fields for token generation* on page 413

Select a database table and columns for token generation

When you choose to use a database source for attributes, you follow this path through the configuration steps.

On this screen you begin to specify exactly where additional data can be found to complete the token-generator contract when you send to this SP (see *Select an attribute retrieval method for token generation* on page 408). Only one table may be used as a source of data for a JDBC lookup.

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 *http://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html*.

Field descriptions

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 a JDBC query. Other databases, such as MySQL, do not require selection of a schema.
Table	The name of the table contained in the database. Use the drop-down to change the table.
Columns to return from SELECT	Displays selected table columns. Select the columns that are associated with the desired attributes you would like to return from the JDBC query.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Database Table and Columns** under the Token Generator Mapping & User Lookup tab.

To select a database table and columns for queries:

1. Choose a Schema file (when applicable) from the drop-down list.
2. Choose a Table from the drop-down list.
3. Choose a name under Columns to Return from Select and click **Add Attribute**.

   Tip: Click **Refresh** if you are updating an existing configuration and changes may have been made to the database.

   Repeat this step for other columns as needed.

   Note: You do not need to add a column here for it to be used as part of a search filter (see *Specify a database filter for token creation* on page 395 next). Add only attributes from which you need actual values to pass in a token.

   Tip: To determine what attributes to look up during a query, click the **View Attribute Contract** link to see what information must be collected (see *Define an attribute contract for SP STS* on page 405). Then determine what information is coming in from the token processor (see *Select an attribute retrieval method for token generation* on page 408). Information not contained in the token-processor contract may be pulled from the data store look-up query.

Specify a database filter for token generation

The JDBC `WHERE` clause in PingFederate queries the data table you selected to retrieve a record associated with a particular value (or values) from the incoming security token. The clause is in the form:

```
WHERE column1=value1 [AND column2=value2] [OR ...]
```

The left side of the first variable pair uses a column name in the database table you selected (see Selecting a Data Table and Columns).

The right side generally uses values passed in from the incoming SAML token (variables, including the correct formatting, are listed under Assertion Values).

You can also apply additional search criteria from your own database, using any other columns from the targeted table.

Tip: Click "**View List of Columns . . .**" to see a list from which to copy and paste.

For more information about `WHERE` clauses, consult your DBMS documentation.

EXAMPLE:

```
userid='${username}'
```

In this example `userid` is the name of a column in the JDBC data store. On the right side, `'${username}'` returns the value of the `username` variable from the IdP token processor.

Important: You *must* use the `${}` syntax to retrieve the value of the enclosed variable and use single quotation marks around the `${}` characters.

Field description

Field	Description
Where	`WHERE` clause statements conditionally select data from a table. Enter the `WHERE` clause statement in the space provided. For example: `WHERE email='clive@company.com'`.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Database Filter** from the steps list under the Token Generator Mapping & User Lookup tab.

To construct the `WHERE` clause:

1. Enter the statement in the space provided, following the guidelines and example above.

   The initial `WHERE` is optional.
2. Ensure the syntax and variable names are correct.

When you click **Next**, you will map attribute values returned from the database into the security token (see *Fulfill the attribute contract for token generation* on page 413).

Configure an LDAP search for token generation

When you choose to use an LDAP source for attributes, you follow this path through the configuration steps.

On this screen you specify the branch of your LDAP hierarchy where you want PingFederate to look up user data.

Field Descriptions

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.

Field	Description
Search Scope	Determines the node depth of the query. Select Subtree, One level or Object.
Root Object Class	The class containing the attributes you want.
Attributes to return from search	A list of added from the drop-down list below. Subject DN is a default attribute, which may be used as the primary user identifier.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **LDAP Directory Search** under the Token Generator Mapping & User Lookup tab.

To select LDAP attributes:

1. Optional: Enter a Base DN.
2. Select a Search Scope.
3. Select a Root Object Class.
4. Under Attributes to return from search, choose an attribute and click **Add Attribute**.

   Note that the attribute Subject DN is always returned by default.

   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 as well as indirectly through nested group membership (if any) under the Base DN.

   For example, suppose you have three groups under the Base DN, namely 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).
5. Repeat the last step for other attributes as needed.

   Note: You do not need to add an attribute here for it to be used in a search filter (see *Specify an LDAP filter for token generation* on page 412). Add only attributes from which you need actual values to pass into the outgoing security token.

Specify an LDAP filter for token generation

The LDAP filter queries the data you selected to retrieve a record associated with a particular value (or values) from the incoming token. The filter is in the form:

```
attribute=${value}
```

The left-side variable is an attribute you selected earlier (see Configuring LDAP Search Parameters).

The right side generally uses values passed in from the incoming SAML token (variables, including the correct syntax, are listed under Assertion Values).

You can also apply additional search criteria from your data store, using any other attributes from the targeted object classes.

Tip: Click "**View List of Available LDAP Attributes**" for a list from which you can copy and paste.

For general information about search filters, consult your LDAP documentation.

Field descriptions

Field	Description
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: 1) at least one attribute (attribute data type) to search on, 2) a search filter operator that will determine what to match, and 3) the value of the attribute being sought. Searches must have at least one of each of these three components.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.
7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **LDAP Filter** under the Token Generator Mapping & User Lookup tab.

To construct the LDAP filter:

1. Enter the statement in the space provided, following the guidelines and example above.

   Note: If you used an anonymous binding to create this LDAP connection, your access might be restricted (see *Configure an LDAP connection* on page 145).
2. Ensure the syntax and variable names are correct.
3. Click **Next**.

Configure custom source filters for token generation

When you choose to use a custom source for attributes, you follow this path through the configuration steps.

On this screen you specify a filter, or lookup 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.

Select custom source fields for token generation

On the Configure Custom Source Fields screen, you can choose from among the fields shown to map to the token processor contract. These choices are supplied by the driver implementation. Select only those needed to fulfill the attribute contract for this partner connection.

Fulfill the attribute contract for token generation

You map attributes for outgoing security tokens for this partner on the Token Generator Contract Fulfillment screen.

Map each attribute to fulfill the Token Generator Contract from one of these Sources:

- Assertion

  When you make this selection, the associated Value drop-down list is populated by the incoming SAML token (Assertion).
- Context

  Values are returned from the context of the transaction at runtime.

  Note: The HTTP Request and STS SSL Client Certificate Chain selections are retrieved as 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 (see *Using the OGNL edit screen* on page 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)

  Note: When using the STS Basic Authentication Username, STS SSL Client Certificate's Subject DN, or STS SSL Client Certificate Chain attributes, ensure the associated authentication is enabled and configured in WS-Trust STS Settings (see *Select authentication methods* on page 379).
- LDAP/JDBC/Custom

  Note: PingFederate appends a description in parentheses for configured data store lookups (see *Configure attribute sources and user lookup for token creation* on page 392).

  Values are returned from your attribute source (if you are using data store—see *Select an attribute retrieval method for token creation* on page 391). When you make this selection, the Value list is populated by the LDAP, JDBC, or Custom attributes you identified as an Attribute Source (see *Configure an LDAP search for token creation* on page 396, *Select a database table and columns for token creation* on page 394, or *Configure custom source filters for token creation* on page 398).
- 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 (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 assertion, 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 data store attributes you select.

To reach this screen for editing:

1. Click **SP Configuration** on the Main Menu.
2. Click the connection name under IdP Connections.

   Click **Manage All**, if needed, to see a full list of connections.
3. Click **WS-Trust STS** under the IdP Connection tab.
4. Click **Configure WS-Trust STS**.
5. Click **Token Generation** under the WS-Trust STS tab.

   If this step is not shown, token generation is not selected for the connection (see *Configure protocol settings for SP STS* on page 405).
6. Click **Configure Token Generation**.

7. Click **Token Generator Mapping & User Lookup** on the Summary screen.
8. Click the Token Generator Instance Name.
9. Click **Token Generator Contract Fulfillment** under the Token Generator Mapping & User Lookup tab.

To map attributes:

1. Choose a Source for each Target attribute.
2. Choose (or enter) a Value for each Attribute.

   See *Map each attribute to fulfill the Token Generator Contract from one of these Sources:* above. All values must be mapped.
3. Click **Next**.

Specify issuance criteria for token generation (Optional)

Use this screen to define criteria PingFederate can evaluate to determine whether to issue a security token for a user (see *About token authorization* on page 71). This token authorization can be used to restrict who can access identity-enabled Web Services.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Assertion – Select to access attributes from the assertion.
   - Context – Select to use values returned from the context of the transaction at runtime.

     Note: The HTTP Request and STS SSL Client Certificate Chain selections are retrieved as Java objects rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values. If you want to use an expression for authentication, skip to Step 8 below.
   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for (see *Configure attribute sources and user lookup* on page 265).
   - Mapped Attributes – Select to access the required attributes.
2. Select an attribute name.

   Note: When using the STS Basic Authentication Username, STS SSL Client Certificate's Subject DN, or STS SSL Client Certificate Chain attributes, ensure the associated authentication is enabled and configured in WS-Trust STS Settings (see *Select authentication methods* on page 379).
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

   Important: When using the STS SSL Client Certificate's Subject DN attribute, you must select one of the following conditions: equal to DN, not equal to DN, multi-value contains DN, or multi-value does not contain DN. These operators normalize the DN before comparison to accommodate for different string representations that are still considered equivalent (for example, case sensitivity, or whitespace).
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in a `SOAP` message returned. 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`).

Note: 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.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Enable and disable expressions* on page 485).

Important: When you multiplex one connection for multiple environments (see *Connecting to a partner in one connection* on page 81), consider using an OGNL expression to verify the virtual server ID in conjunction with other conditions, such as group membership information, to protect against unauthorized access (see *Sample OGNL expressions* on page 487).

- Use the in-line editor box to enter the OGNL expression.

  For more information about OGNL, see *Attribute mapping expressions* on page 485.
- Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

  Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.
- Click **Add**. Continue to add expressions, as needed.
- Click the **Test** link to input values and test the resulting output for the expression.

  Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

Review the SP token generator mapping

When you have finished configuring Token Generator Mapping & User Lookup, 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.

- If you are editing an existing connection, click **Done** on successive screens until you reach the WS-Trust STS screen, and then click **Save**.

To save a new configuration:

1. Click **Done** to return to the Token Generator Mapping & User Lookup screen.
2. Click **Next** to go to the Token Generation Summary screen, and then click **Done**.
3. On the Token Generation screen, click **Done**.
4. On the WS-Trust STS screen, click **Save**.

Review the token generation configuration

When you have finished configuring Token Generation, 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.

- If you are editing an existing connection, click **Done** on successive screens until you reach the WS-Trust STS screen, and then click **Save**.

Review the SP STS configuration

On the WS-Trust STS Summary screen, you can review the configuration for this connection.

- If you need to make any changes to a new or existing connection, click the heading over the information you want to modify.

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 417
- *Token translator mappings* on page 421
- *Deploying PingFederate as a federation hub* on page 86

Note: The information in this chapter is presented from the viewpoint of an administrative user with "Admin" permissions (see *Account management* on page 105).

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 78).

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 238
- *Manage SP adapters* on page 309

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 495).

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.

Identify target application (optional)

Use this screen to 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*.) Both fields are optional.

To complete the configuration:

- Enter an Application Name and an Application Icon URL.

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 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable*

expressions on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)

- 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 418).

- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 485). 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 *Choosing a Data Store (optional)*) 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 315).

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 452).

Configure issuance criteria (optional)

Use this screen to define criteria PingFederate can evaluate to determine whether to issue an SP adapter token for a user (see *About token authorization* on page 71). This token authorization can be used to restrict who can SSO to protected resources.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Adapter – Select to access attributes from the IdP Adapter.
   - Context – Select to use values 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. If you want to use an expression for authentication, skip to Step 8 below.

   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

Note: PingFederate appends a description in parentheses for configured data store lookups (see *Configure attribute lookup for adapter-to-adapter mapping* on page 418)

- Mapped Attributes – Select to access the required attributes.

2. Select an attribute name.
3. Select the Condition you want to apply.

Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

4. Enter an exact value for the attribute.

Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).

5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Error results are handled in one of two ways:

   - **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 (for more information on this and other user-facing templates, see *Customizable user-facing screens* on page 120).

     Note: Using an error code in the Error Result field allows the error template or a web application to process the error result in a variety of ways—for example, a redirect to another page or e-mail to an administrator.

   If you leave this field blank, a default ACCESS_DENIED error result is used if the authorization fails.
6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

Note: Expressions must be enabled for the **Show Advanced Criteria** button to appear (see *Enable and disable expressions* on page 485).

- Use the in-line editor box to enter the OGNL expression.

  For more information about OGNL, see *Attribute mapping expressions* on page 485.
- Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

  Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.
- Click **Add**. Continue to add expressions, as needed.
- Click the **Test** link to input values and test the resulting output for the expression.

  Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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 57). 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 STS* on page 378). 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 381
- *Manage token generators* on page 402

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** on the Main Menu.
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 452).

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 495).

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 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485. For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)
- 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 *Choosing a Data Store for Token Mapping*).
- Expression (when enabled)

  This option provides more complex mapping capabilities—for example, transforming incoming values into different formats (see *Attribute mapping expressions* on page 485). 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 *Data Store Configuration for Token Mapping*) 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).

Specify token issuance criteria

Use this optional configuration to define criteria PingFederate can evaluate to determine whether to issue a security token (see *About token authorization* on page 71). This token authorization can be used to restrict who can access protected Web Services.

To configure Issuance Criteria:

1. Select a source containing attributes for token authorization.

   Associated attributes appear in the Attribute Name drop-down list:

   - Context – Select to use values returned from the context of the transaction at runtime.

     Note: The HTTP Request and STS SSL Client Certificate Chain selections are retrieved as Java objects rather than text. For this reason, OGNL expressions are more appropriate to evaluate and return values. If you want to use an expression for authentication, skip to Step 8 below.

     Choose Expression and then click **Edit** to enter an expression (see *Using the OGNL edit screen* on page 488). (If the Expression selection is not listed, then the feature is not enabled—see *Enable and disable expressions* on page 485.

     For syntax and examples, see sections under *Construct OGNL expressions* on page 486.)
   - LDAP/JDBC/Custom (if a data store is configured) – Select to access attributes returned from a data source.

     Note: PingFederate appends a description in parentheses for configured data store lookups (see *Choosing a Data Store for Token Mapping*).
   - Mapped Attributes – Select to access the required attributes from the token contract.
   - Token – Select to access attributes from the IdP token processor.
2. Select an attribute name.
3. Select the Condition you want to apply.

   Note: Use multi-value conditions when you want PingFederate to verify that an attribute contains or does not contain the specified value in its attribute list. Using a single-value condition when an attribute has multiple values causes the condition to evaluate consistently to false.

   Important: When using the STS SSL Client Certificate's Subject DN attribute, you must select one of the following conditions: equal to DN, not equal to DN, multi-value contains DN, or multi-value does not contain DN. These operators take into account minor differences that might be present in the DN syntax (for example, whether spaces are present after commas).
4. Enter an exact value for the attribute.

   Tip: You can use OGNL expressions to perform more complex evaluations, including partial matching (see step 8 for more information).
5. Optional: Use the Error Result field to customize an error code or error message for use if the token authorization fails.

   Errors result in a `SOAP` message returned. 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`).

   Note: 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, the user sees a default ACCESS_DENIED error result at runtime if the authorization fails.

6. Click **Add**.
7. Repeat the steps above to add additional attributes, as needed, for each authorization criteria.

   Note: All criteria must pass in order for a user to be authorized.

8. Optional: Click **Show Advanced Criteria** to configure more complex evaluations using OGNL expressions.

   Note: Expressions must be enabled for the Show Advanced Criteria button to appear (see *Enable and disable expressions* on page 485).

   - Use the in-line editor box to enter the OGNL expression.

     For more information about OGNL, see *Attribute mapping expressions* on page 485.
   - Use the Error Result box to enter an error code or an error message for use if authorization fails (see step 5 above for more information).

     Note: If the OGNL expression resolves to a string value (rather than true or false), then the returned value overrides the Error Result.
   - Click **Add**. Continue to add expressions, as needed.
   - Click the **Test** link to input values and test the resulting output for the expression.

     Note: For more information on testing OGNL expressions, see *Using the OGNL edit screen* on page 488.

9. Click **Next** (or **Done**, if you are finished with the configuration, and then **Save** on the Manage Mappings screen).

To edit Issuance Criteria:

- Click **Edit** under Actions for the criteria.

To delete Issuance Criteria:

- Click **Delete** under Actions for the criteria and then click **Save**.

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:

- *Kerberos Adapter* on page 424
- *HTML Form Adapter* on page 428
- *HTTP Basic Adapter* on page 432
- *OpenToken Adapter* on page 434
- *Composite Adapter* on page 439

Kerberos Adapter

The PingFederate 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 (see the *User Guide* for IWA Integration Kit 3.1 for more information). 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 workflow 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 (see *About token authorization*).

Note: The `SIDs` attribute contains multiple values. Use the `multi-value contains` condition (or `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 (see *Attribute mapping 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* (`technet.microsoft.com/en-us/library/dd378897%28v=ws.10%29.aspx`) from Microsoft.

Configure the Kerberos Adapter

1. If you have not already done so, log on to the PingFederate administrative console.
2. Click **IdP Configuration** on the Main Menu.
3. Click **Adapters** under Application Integration Settings.
4. On the Manage IdP Adapter Instances screen, click **Create New Instance**.
5. On the adapter Type screen, enter an **Instance Name** and **Instance Id**, select `Kerberos Adapter` as the Type, and click **Next**.

   The **Instance Id** may not contain spaces or underscores.
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**.

   Note: If this is a child instance, select the override check box related to the settings you want to modify.
8. 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 179).
9. 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 a 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 (see *Composite Adapter configuration*).

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. (For more information, see *Customizable user-facing screens*.)

10. 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 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. For more information on Velocity templates, see *Customizable user-facing screens*.
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.)

11. Click **Next**.
12. On the Adapter Attributes screen, select **Username** (and optionally **Domain/Realm Name**) to be used in constructing a unique identifier (Pseudonym) for account linking (see *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.

You can also choose to mask the values of any or all attributes that PingFederate logs from the adapter at runtime (see *Attribute masking*).

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 (see *Attribute mapping expressions*).

13. Click **Next**.
14. On the Summary screen, click **Done**.
15. 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 427
- *Configure Mozilla Firefox* on page 427

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 `sso.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, `sso.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, `sso.example.com`).
5. Click **OK** and close the `about:config` tab.
6. Optional: Exit Firefox.

HTML Form Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or IdM system logon module. PingFederate's adapter and application agents are typically used to integrate with these local authentication mechanisms.

PingFederate packages an HTML Form Adapter that delegates user authentication to a configured password credential validator. This authentication mechanism validates credentials based on either an LDAP directory or a simple username validator that authenticates credentials maintained by PingFederate.

On the IdP side, when the PingFederate IdP server receives an authentication request for SP-initiated SSO or the user clicks a link for IdP-initiated SSO, the IdP server invokes the HTML Form Adapter and, if not already authenticated, prompts the user for local IdP credentials. The credentials are then validated using the designated password credential validator and, if validated, a SAML assertion is generated.

The HTML Form Adapter allows you to customize a different login page for each configured adapter instance. You can define a logout path and page or a logout redirect page. You can also enable users to change their network passwords and customize a change-password page, or redirect users to a company-hosted password management system.

PingFederate tracks login attempts per adapter instance. This capability adds a layer of protection against brute force and dictionary attacks. When the **Challenge Retries** threshold is reached, the user is locked out for one minute.

The default value for **Challenge Retries** is three. If you prefer a higher value, consider reviewing the account lockout policy of your directory servers first. For example, if the directory server's account lockout threshold is set to five and the **Challenge Retries** in the HTML Form Adapter is also set to five (or higher), the fifth single sign-on attempt could potentially lock the user accounts on the directory server.

The lockout period is customizable by modifying the **LockoutPeriod** value in `<pf_install>/pingfederate/server/default/data/config-store/com.pingidentity.common.security.AccountLockingService.xml`. This lockout is per adapter and is unlocked after the lockout period has lapsed.

Note: For a PingFederate cluster environment, modify the `com.pingidentity.common.security.AccountLockingService.xml` file on the console node and use the **Server Configuration** > **Cluster Management** > **Replicate Configuration** workflow in the administrative console to replicate the configuration to all engine nodes.

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 **IdP Configuration** > **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, 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 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 defined password credential validators is able to authenticate the user's credentials, and the challenge retries maximum has been reached, the process fails.

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 one minute. 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** (the default choice) is selected, a session is tied to a specific adapter instance. When **None** is selected, a session is not maintained.
Session Timeout	The number of idle minutes before a session times out based on inactivity. The default value is `60` (an hour). If left blank, the lifetime falls back on the **Session Max Timeout** field value. Ignored if **None** is selected for the **Session State** field.
Session Max Timeout	The maximum lifetime (in minutes) before a session expires regardless if the **Session Timeout** field value has been reached, or not. The default value is `480` (8 hours). Ignored if **None** is selected for the **Session State** field. **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.

Property	Description
	Tip: Session information is stored in the PF cookie. By default, the PF cookie is a session cookie; web browsers typically remove session cookies on exit. For more information, see *Extend the lifetime of the PF cookie* .
Login Template (Required)	The template file on the IdP server that displays when prompting the users for their credentials. PingFederate allows each configured adapter instance to use a different login page template. The default template file, `html.form.login.template.html`, is 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.
Logout Redirect	The landing page at the SP after successful IdP logout (applicable only when the **Logout Path** field is configured).
Logout Template	The template file on the IdP server to display after successful IdP logout, when the **Logout Path** field is configured but the **Logout Redirect** field is not.
Allow Password Changes	Enables or disables the ability for users to change their network passwords using this adapter instance. Select the check box to enable the change password functionality. **Note:** The LDAP Password Credential Validator (PCV) and the PingOne directory PCV are currently the only PCVs bundled with PingFederate that support the change password feature.
Change Password Template	The template file on the IdP server that displays when prompting users to change their password. PingFederate allows each configured adapter instance to use a different change password template.
Change Password Message Template	The template file on the IdP server that notifies users their password was successfully changed.
Password Management System	The URL for redirecting users to a company-specific password management system to change their password.
Password Management System Message Template	The template file on the IdP server that notifies users they are being redirected to a password management system to change their password.
Login Challenge Template	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.
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.

Property	Description
	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.
'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 default is 30. 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.
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.
Track Authentication Time	When selected (the default), 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.
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).

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 workflow to validate one or more criteria prior to the issuance of the adapter contract.
8. On the **Summary** screen, review your configuration, modify as needed, and click **Done** to exit the **Create Adapter Instance** workflow.
9. On the **Manage IdP Adapter Instances** screen, click **Save** to retain the configuration of the adapter instance.

   If you want to exit without saving the configuration, click **Cancel**.

HTTP Basic Adapter

Initial user authentication is normally handled outside of the PingFederate server using an application or IdM system logon module. PingFederate's adapter and application agents are typically used to integrate with these local authentication mechanisms.

PingFederate packages an HTTP Basic Adapter that delegates user authentication to a configured password credential validator. This authentication mechanism validates credentials based on either an LDAP directory or a simple username validator that authenticates credentials maintained by PingFederate.

On the IdP side, when the PingFederate IdP server receives an authentication request for SP-initiated SSO or the user clicks a link for IdP-initiated SSO, the IdP server invokes the HTTP Basic Adapter and, if not already authenticated, prompts the user for local IdP credentials. The credentials are then validated using the designated password credential validator and, if validated, a SAML assertion is generated.

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 **IdP Configuration** > **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, 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 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 defined password credential validators is able to authenticate the user's credentials, and the challenge retries maximum has been reached, the process fails.

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 workflow 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**.

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 *www.pingidentity.com*. 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 *www.pingidentity.com* 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.

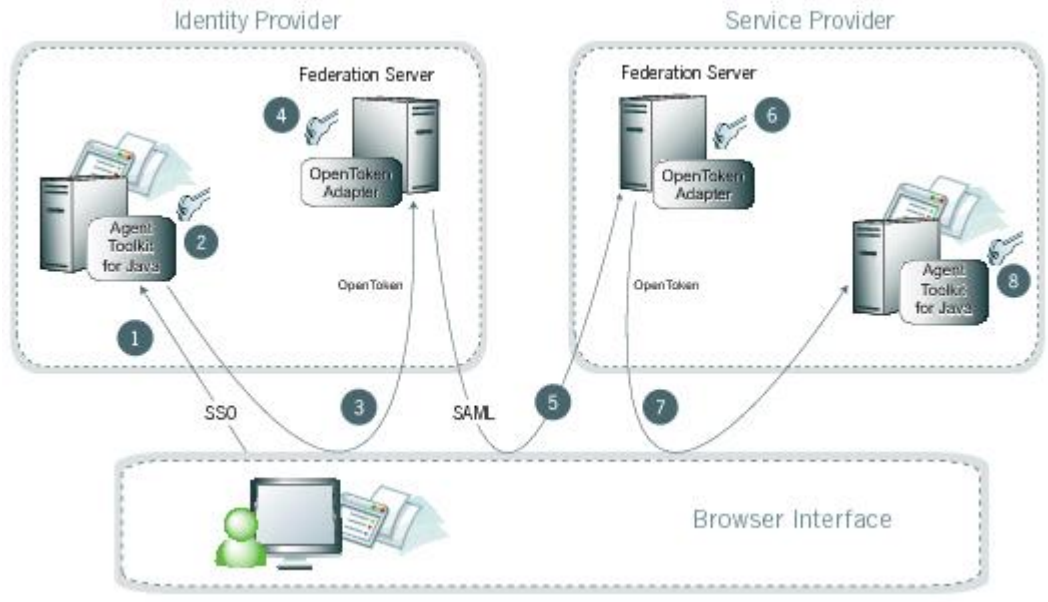


Figure 25: SP-Initiated SSO: POST/POST

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 **IdP Configuration** > **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 referred 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**.

Field	Description
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.
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 workflow 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 **SP Configuration** > **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** 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 **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 referred as the *shared secret*.

Click **Show Advanced Fields** to review the following settings.

Field	Description
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.
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.

Field	Description
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** to exit the **Create Adapter Instance** workflow.
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 policy.

Tip: For complex authentication requirements, consider implementing one or more authentication policies in the **IdP Configuration** > **Policies** screen. For more information, refer to *Authentication policies*.

Configure the Composite Adapter

1. Click **IdP Configuration** > **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.

Column	Description
	Important: For the sufficient policy to work correctly, the adapter must return control to PingFederate after any kind of a failure. Currently, not all types of adapters have this capability. For an up-to-date list of supported adapters, see *Known issues and limitations*.
AuthN Context Weight (Required)	If more than one adapter instance in the chain is capable of returning an authentication context, this relative weight is used to determine which value is included in the assertion, *unless* the value is overridden under **AuthN Context Override**. If weights are the same for two or more contexts, the first one processed is included in the assertion. The default value is `3`.
AuthN Context Override	If provided, this value overrides the authentication context value that may be returned from the adapter instance. The value in this field may be sent in the assertion if the associated adapter instance is invoked. If weights are the same for two or more contexts, the first one processed is included in the assertion.

c) Add at least one more adapter instance and configure its **Policy**, **AuthN Context Weight**, and **AuthN Context Override** settings.

Repeat this step to add more adapter instances as needed.

At runtime adapter chaining is sequential, starting at the top of the list.

Important: Several types of adapters—for example, the Integrated Windows Authentication 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.

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 (see *Configure the HTML Form Adapter*).

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 workflow 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**.

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 120).

SP endpoints also include those available for SCIM Inbound Provisioning (see *Provisioning for SPs* on page 79).

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 417) and retrieval of WS-Trust metadata (see *WSC and WSP support* on page 58).

PingFederate provides a favorite icon for all Application Endpoints. For more information, see *Customize the favicon for application and protocol endpoints* on page 128.

IdP endpoints

The following sections describe PingFederate IdP 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 IdP server, for example:

```
https://sso.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`).

Parameters are case-sensitive.

/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 465.

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 is case-sensitive.
	One of these parameters is required unless the federation ID can be derived from `TargetResource` or `TARGET`.

TargetResource or TARGET	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.

Parameter	Description
(optional)	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 (see *Customizable user-facing screens* on page 120).

`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 (see the next parameter, `ACSIdx`).

`ACSIdx` (optional - SAML 2.0)	Specifies the index number of partner's ACS (see *Set Assertion Consumer Service URLs (SAML)* on page 279). 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 selection configuration. For example, the CIDR authentication selection could enforce the use of a given adapter based on whether a user is on or off the network (see *Manage authentication selectors* on page 208).
`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 253) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 137).

/idp/startSLO.ping

This is the path used to initiate an IdP-initiated SLO (under SAML 2.0) or an OpenID Connect logout (see *Asynchronous front-channel logout* on page 183). Typically, a systems integrator or developer creates one or more links to this endpoint in the protected resources of their IdP application or portal to allow users to end their sessions at various SPs. This endpoint uses the local PingFederate session to determine which SPs have been issued an SSO assertion and sends them a SAML logout request.

The following table shows the HTTP parameters for this endpoint.

Parameter	Description
`TargetResource` (optional)	Indicates where the user is redirected after a successful SLO. If this parameter is not included in the request, PingFederate uses as a default the URL for a successful SLO as entered on the IdP Default URL screen. Note that the parameter value must be URL-encoded.
`InErrorResource` (optional)	Indicates where the user is redirected after an unsuccessful SLO. If this parameter is not included in the request, PingFederate redirects the user to the SLO error landing

Parameter	Description
	page hosted within PingFederate (see *Customizable user-facing screens* on page 120).

`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 282).

/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.

System-services endpoints

These endpoints are the same for both IdPs and SPs and are described under *System-services endpoints* on page 452.

SP endpoints

The following sections describe the PingFederate SP endpoints, including the query parameters that each accepts or requires:

- *SP services* on page 444
- *SCIM inbound provisioning endpoints* on page 449

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://sso.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 465.

The following table shows the HTTP parameters for this endpoint.

Note: Some parameters described below can have multiple values. Specify these values by using multiple independent query string parameters of the same name.

Parameter	Description
`PartnerIdpId`	The federation ID of the IdP that authenticates the user and issues an assertion. This ID is case-sensitive. Required if more than one IdP connection is configured and `Domain` is not used, and SP authentication policies are turned off. Not required if SP authentication policies are turned on.
`SpSessionAuthnAdapterId`	The explicit SP adapter instance ID indicating the adapter to use to create an authenticated session or security context. Optional if SP authentication policies are turned off. Required if SP authentication policies are turned on, unless the PingFederate SP server is able to determine the applicable SP adapter instance based on the target URL mapping configuration and the `TargetResource` or `TARGET` value at runtime.

`TargetResource`
 	
or
 	
`TARGET`
	This parameter indicates where the end-user is redirected after a successful SSO (the target applications).
	
	Note that the parameter value must be URL-encoded.
	
	When this parameter is not provided in the URL, a default target resource may be specified in the administrative console, either for all IdP connections (see *Configure default URLs* on page 315) or for individual connections (see *Configure default target URLs (optional)* on page 351), 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 120).

`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 346).

`AllowCreate`	Controls the value of the `AllowCreate` attribute of the `NameIDPolicy` element in the AuthnRequest. (The default is `true`.)

Parameter	Description
(optional - SAML 2.0)	
`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)	This parameter controls the attribute of the same name in the AuthnRequest. (The default is`false`.)
`IsPassive` (optional - SAML 2.0)	This parameter controls the attribute of the same name in the AuthnRequest. (The default is `false`.)
`RequestedACSIdx` (optional - SAML 2.0)	The index number of your site's Assertion Consumer Service, where you want the assertion to be sent.
`RequestedAcsUrl` (optional - SAML 2.0)	The URL of your site's Assertion Consumer Service, where you want the assertion to be sent.
`RequestedAuthnCtx` (optional - SAML 2.0)	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*). 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 305).
`vsid` (optional)	Specify the virtual server ID. When absent, PingFederate uses the default virtual server ID (if specified) for the connection (see *Identify the IdP* on page 324) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 137).
`Domain`	The domain name associated with the requesting user's IdP to invoke Auto-Connect. In this case, `PartnerIdpId` cannot be used (see *Using Auto-Connect* on page 75).

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 311).

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 282).

`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 120).
`SpSessionAuthnAdapterId` (optional)	The SP adapter instance ID indicating which session to terminate and which IdP will receive the logout request.
`SourceResource` (optional)	A URL indicating the origin of the logout request. It is mapped to an adapter ID in order to designate which session to terminate.

An SP PingFederate session can be associated with one or more application sessions relying on any number of IdPs as the session authority. PingFederate must choose one session to terminate and also send an SLO request to the IdP that issued the assertion that created the session. Sessions are associated with the ID of the adapter instance that created them. Once an adapter ID is determined, the first session found with that ID is used. Determination of the adapter instance ID occurs in the following order:

1. If there is a value for the `SpSessionAuthnAdapterId` parameter, it is used.
2. If there is a value for the `SourceResource` parameter, PingFederate attempts to map a URL to an adapter using that value to determine the adapter ID.
3. If there is an HTTP header value for `Referer` [sic], PingFederate attempts to map a URL to an adapter using that value to determine the adapter ID.
4. If none of the above is successful, the `TargetResource` parameter value or the value for the default SLO success URL are used to map a URL to an adapter.
5. Finally, if no adapter ID is determined, the first one in the list is used.

/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 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* in the "Supported Standards" chapter of the *Get started with PingFederate* guide). 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 445). 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* in the "Supported Standards" chapter of the *Get started with PingFederate* guide).

The following table shows the HTTP parameters for this endpoint.

Note: Some parameters described below can have multiple values. Specify these values by using multiple independent query string parameters of the same name.

Parameter	Description
`Subject`	Uniquely identifies the user to the IdP. When user authenticates with an X.509 certificate, this is the Subject DN, which must be URL-encoded.
`Issuer` (optional)	The IssuerDN from the user's X.509 certificate (when XASP is used), which uniquely identifies the entity that issued the user's certificate. The parameter must be URL-encoded. **Note:** When specified this parameter overrides the Subject parameter.
`PartnerIdpId` (except for XASP)	Used to identify the specific IdP partner to which the Attribute Query should be sent. If this parameter is not present, the Subject and Issuer are used to determine the correct IdP. **Note:** For XASP, this parameter overrides both the Subject and Issuer parameters.
`Format` (required for XASP, otherwise optional)	Identifies the name-identifier format of the Subject query parameter. If included, the value must be one of the SAML 2.0 Name Identifier Format URIs (see section 8.3 of the *SAML specifications* (`docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`). **Note:** For XASP, this parameter must be set to `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`. If not specified, the parameter defaults to `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`. Note that the parameter must be URL-encoded.
`AppId`	The unique identifier of the initiating application.

Parameter	Description
`SharedSecret`	Used to authenticate the initiating application. The AppId and SharedSecret must both match the application authentication settings within the PingFederate server.
`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 253) or the SAML federation ID defined in Server Settings (see *Specify federation information* on page 137).

SCIM inbound provisioning endpoints

The following sections describe the PingFederate endpoints for SCIM inbound provisioning (see *Inbound provisioning* on page 79).

Note: Begin each URL with the fully qualified server name and port number of your PingFederate server, for example:

```
https://pingidentity.com:9031/pf-scim/v1/Schemas
```

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`)

Note: The PingFederate implementation is described in the next couple sections, */pf-scim/v1/Users* on page 449, */pf-scim/v1/Groups* on page 450, and */pf-scim/v1/Schemas* on page 452. Developers can find additional information about the implementation via the Service Provider Configuration Endpoint (see */pf-scim/v1/ServiceProviderConfigs* on page 452).

/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 (see *Configure inbound provisioning* on page 359). JSON is currently the only supported format for the HTTP message body.

HTTP Method	Description
POST	Sends user attributes in JSON format—defined in the SCIM Core Schema—to create a new user. If the user is successfully provisioned, the HTTP response indicates a 201 status code and contains a JSON body indicating all of the user attributes added to the datastore. The user ID is set as the id attribute in the JSON response, and the full URL to reference the user is in the HTTP response Location header. For an existing user, you can also use POST either to update or delete/disable user record by appending the user ID to the path (see the Note below) 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 below.)
GET	`/pf-scim/v1/Users`

HTTP Method	Description

	• Retrieves all attributes from all users.
	• A successful response is indicated by an HTTP 200 status code and a list of all users and their attributes.
	
	`/pf-scim/v1/Users/<user_id>`
	
	• Retrieves all attributes for the specified user (by the person's user ID value).
	• A successful response is indicated by an HTTP 200 status code and the attributes.
	
	`/pf-scim/v1/Users?attributes=<attribute>`
	
	• Retrieves the specified attribute from all users.
	• A successful response is indicated by an HTTP 200 status code along with a list of the specified 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 the 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`).
	
	Tip: You can use both the `attributes=<attribute>` and `filter=<filter>` query parameters in one query to narrow your search results.

Note: The following methods require a user ID value appended to the endpoint path: `/pf-scim/v1/Users/<user_id>`

PUT	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	Deletes or disables the user record for the specified user. Whether a user is deleted or disabled is determined by the connection configuration (see *Configure the handling of SCIM delete requests* on page 366). A successful response is indicated by an HTTP 200 status code.

Note: For a list of HTTP error codes that may be returned, see the *SCIM specifications* (`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 optional. For more information, see *Choose an IdP connection type* on page 322.

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 (see *Configure inbound provisioning* on page 359). JSON is currently the only supported format for the HTTP message body.

HTTP Method	Description
POST	Sends group attributes in JSON format—defined in the SCIM Core Schema—to create a new group. If the group is successfully provisioned, the HTTP response indicates a 201 status code and contains a JSON body indicating all of the group attributes added to the datastore. The group ID is set as the `id` attribute in the JSON response, and the full URL to reference the group is in the HTTP response Location header. For an existing group, you can also use POST either to update or delete the group by appending the group ID to the path (see the Note below) 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 below.)

GET	`/pf-scim/v1/Groups`
	
	• Retrieves all attributes from all groups.
	• 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>`
	
	• Retrieves all attributes for the specified group (by its group ID value).
	• A successful response is indicated by an HTTP 200 status code and the attributes.
	
	`/pf-scim/v1/Groups?attributes=<attribute>`
	
	• Retrieves the specified attribute from all groups.
	• A successful response is indicated by an HTTP 200 status code along with a list of the specified 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 the 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`).
	
	Tip: You can use both the `attributes=<attribute>` and `filter=<filter>` query parameters in one query to narrow your search results.

Note: The following methods require a group ID value appended to the endpoint path: `/pf-scim/v1/Groups/<group_id>`

PUT	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.

HTTP Method	Description
DELETE	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 the *SCIM specifications* (`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 (see *Configure inbound provisioning* on page 359). 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.

/pf-scim/v1/ServiceProviderConfigs

This Service Provider Configuration Endpoint is where a client 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.

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 "OK" browser message and an HTTP 200 status indication 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 128).

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 131).

/pf/adapter2adapter.ping

This endpoint initiates direct IdP-to-SP adapter mapping, when that feature is configured (see *Adapter-to-adapter mappings* on page 417).

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 uses as a default the URL for a successful SSO (see *Configure default URLs* on page 315).
`SpSessionAuthnAdapterId` (optional)	The SP adapter instance ID to be used. If not provided and more than one SP adapter instance is configured with adapter-to-adapter mapping, PingFederate uses configured defaults (see *Configure target URL mapping* on page 311).
`IdpAdapterId` (optional)	Indicates the IdP adapter to use for authentication if more than one IdP adapter is configured in adapter-to-adapter mappings.
`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 120).

/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 (see *Token translator mappings* on page 421).

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.

/pf/sts_mex.ping

This endpoint returns STS metadata for use in expediting configuration of Web- service applications.

The following table shows the HTTP parameters for this endpoint:

Parameter	Description
`PartnerSpId`	The Connection ID of the SP to whom the SAML token will be issued. This parameter determines the connection for which metadata will be generated.
`PartnerIdpId`	The Connection ID of the IdP issuing the SAML token to be consumed by PingFederate. This parameter determines the connection for which the metadata will be generated.
`vsid` (optional)	Specify the virtual server ID. If absent, PingFederate uses the default virtual server ID (if specified) for the connection or the federation ID defined in Server Settings (see *Specify federation information* on page 137).

/pf/federation_metadata.ping

This endpoint returns SAML and WS-Federation metadata.

The following table shows the HTTP parameters for this endpoint:

Parameter	Description
`PartnerSpId`	The Connection ID of the SP to whom the assertions or tokens are issued. This parameter determines the connection for which metadata is generated.
`PartnerIdpId`	The Connection ID of the IdP issuing the assertions or tokens to be consumed by PingFederate. This parameter determines the connection for which the metadata is generated.
`vsid`	Specify the virtual server ID.
(optional)	If absent, PingFederate generates the metadata based on the connection's default virtual server ID (if two or more virtual server IDs are defined) or the federation ID defined in the **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.

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 *RFC4648* (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 examples, 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* (tools.ietf.org/html/rfc6749) and (if applicable) *the OpenID Connect specification* (openid.net/connect), 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 resource owners to revoke authorization grants, and the clients to retrieve OpenID Connect metadata, such as the User Info endpoint.

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: Begin each URL with the fully qualified server name and port number of your IdP or SP PingFederate server; for examples:

- `https://sso.example.com:9031/as/authorization.oauth2`
- `https://sso.example.com:9031/as/token.oauth2`

Token endpoint

The token endpoint is defined in the OAuth 2.0 specification 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

Note: Per OAuth specifications, this endpoint accepts only the HTTP POST method.

OAuth client identification and authentication

OAuth clients can authenticate to the OAuth AS using this endpoint by presenting their client identifier and client secret either using the HTTP Basic authentication scheme (where the client identifier is the username, and the client secret is the password) or with the following HTTP request parameters:

Parameter	Description
client_id (Optional)	The client identifier.
client_secret (Optional)	The client secret.

Whenever possible, the use of HTTP Basic is recommended over the use of the request parameters.

Clients without a client secret can use the client_id parameter to identify themselves to the OAuth AS and omit the client_secret parameter.

The only time that the client_id parameter is not required is when the relevant **Unidentified Clients** check box is selected in the **OAuth Settings** > **Authorization Server Settings** screen.

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: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 (Optional)	Required if the authorization request was sent with a code_challenge paramter 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 460.
redirect_uri (Optional)	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 460). The value here must match the authorization-request value, if applicable. The parameter is also required for clients with multiple redirection URIs or one redirection URI that uses wildcards. The parameter is optional for clients with only one specific redirection URI.

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.

Parameter	Description
scope (Optional)	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. The requested scope must be equal to or less than the scope originally 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 Settings** > **Authorization Server Settings** screen. Scopes can also be restricted per client.

Resource owner credentials password 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.
scope (Optional)	The scope of the access request.
validator_id (Optional)	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.

Client credentials grant type

These parameters apply when the grant_type parameter for `/as/token.oauth2` is set to `client_credentials`.

Parameter	Description
scope (Optional)	The scope of the access request.

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 *Section 5 of RFC4648* (tools.ietf.org/html/rfc4648#section-5).
scope (Optional)	The scope of the access request.

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.

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.

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 attributes or name-to-array attributes.

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":"someguy@example.cloud"
}
}
```

OAuth access token management parameters

Parameter	Description
access_token_manager_id (Optional)	The access_token_manager_id value is the instance ID of the desired access token manager. When specified, the PingFederate® AS uses the desired access token management instance for the request if it is eligible; otherwise it aborts the request. **Note:** When the access_token_manager_id parameter is specified, the PingFederate AS ignores the aud parameter.
aud (Optional)	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

Parameter	Description
	match is found, the PingFederate AS 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 preconfigured resource URI. The PingFederate AS takes an exact match over a partial match. If there are multiple partial matches, the PingFederate AS takes the partial match where the provided URI matches more specifically against the preconfigured 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
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 preconfigured 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 preconfigured 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 revocation endpoint

The token revocation endpoint is defined in the OAuth 2.0 Token Revocation (RFC 7009) specification. 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

OAuth clients can authenticate to the OAuth AS using this endpoint by presenting their client identifier and client secret either using the HTTP Basic authentication scheme (where the client identifier is the username, and the client secret is the password) or with the following HTTP request parameters:

Parameter	Description
client_id (Optional)	The client identifier.
client_secret (Optional)	The client secret.

Whenever possible, the use of HTTP Basic is recommended over the use of the request parameters.

Clients without a client secret can use the client_id parameter to identify themselves to the OAuth AS and omit the client_secret parameter.

The only time that the client_id parameter is not required is when the relevant **Unidentified Clients** check box is selected in the **OAuth Settings** > **Authorization Server Settings** screen. In this case, only tokens issued by unidentified clients will be revoked.

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.
token_type_hint (Optional)	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 Settings** > **Authorization Server Settings** screen, the implicit access grant will also be revoked with the access token.

Authorization endpoint

The authorization endpoint is defined in the OAuth 2.0 specification 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 (Optional)	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 (Optional)	Applicable only when `response_type` 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 *Authorization code grant type* on page 456).

code_challenge_method
(Optional)
	Applicable only when the response_type parameter value is `code` and a code_challenge parameter value is provided.
	This parameter indicates the transformation method used to derive the code_challenge parameter value from that of the code_verifier parameter. PingFederate® OAuth AS supports two transformation methods:
	• `plain`, which indicates the code_challenge parameter value is that of the code_verifier parameter.
	• `S256`, which indicates the code_challenge parameter is derived from the code_verifier parameter value as follows:
	`code_challenge=Base64Url-encode(SHA256(ASCII(code_verifier)))`, where:
	  • `ASCII(code_verifier)` denotes the octets of the ASCII representation of the `code_verifier` value.
	  • `SHA256(octets)` denotes the SHA 256-bit hash of the octets.
	  • `Base64Url-encode(octets)` denotes the base64url encoding of octets; the output is URL-safe.
	Note: For detailed information about the transformation method, refer to *Proof Key for Code Exchange (PKCE) by OAuth Public Clients* (tools.ietf.org/html/rfc7636).
	The code_challenge_method parameter value is case-sensitive. An error message is returned to the clients for any other values.
	Note that omitting the code_challenge_method parameter has the same effect as providing the code_challenge_method parameter with a value of `plain`.

redirect_uri	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.

Parameter	Description
	Note that if this parameter is used, the same parameter and value must also be used in subsequent token requests (see *Authorization code grant type*).
claims_locales (Optional)	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 (see *Define grant contract fulfillment for IdP adapter mapping* on page 199 or **OAuth Attribute Mapping** configuration in an IdP connection).
login_hint (Optional)	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 (Optional)	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.
scope (Optional)	The scope of the access request expressed as a list of space-separated, case-sensitive strings. Valid scope values are defined in the **OAuth Settings** > **Authorization Server Settings** screen. **Tip:** Scopes can be restricted per client.
state (Optional)	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 (Optional)	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) (Optional)	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) (Optional)	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 selection configuration. For example, the OAuth Scope Authentication Selector could enforce the use of a given adapter 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 (Optional)	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 (Optional)	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 (Optional)	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 (Optional)	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

Parameter	Description
access_token_manager_id (Optional)	The access_token_manager_id value is the instance ID of the desired access token manager. When specified, the PingFederate® AS uses the desired access token management instance for the request if it is eligible; otherwise it aborts the request. **Note:** When the access_token_manager_id parameter is specified, the PingFederate AS ignores the aud parameter.
aud (Optional)	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, the PingFederate AS 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 preconfigured resource URI. The PingFederate AS takes an exact match over a partial match. If there are multiple partial matches, the PingFederate AS takes the partial match where the provided URI matches more specifically against the preconfigured 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

ATM1	`https://localhost:9031/app1`
	`https://localhost:9031/app2/data`
	`https://app.example.local`

Access Token Management Instances	Resource URIs
ATM2	`https://localhost:9031/app1/data` `https://localhost:9031/app2/data/get`

`https://localhost:9031/app1` (a resource URI preconfigured 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 preconfigured 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.

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) (Optional)	Indicates the entity ID of the connection ID of the IdP with whom to initiate Browser SSO for user authentication.
pfidpadapterid (Optional)	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.

OpenID Connect provider metadata endpoint

This public endpoint provides configuration information for the OAuth clients to interface with PingFederate® using the OpenID Connect protocol.

Endpoint: /.well-known/openid-configuration

The endpoint supports HTTP GET request without authentication or parameter; for example:

```
$ curl -s https://localhost:9031/.well-known/openid-configuration|python -m
 json.tool
```

```
{
    "authorization_endpoint": "https://localhost:9031/as/
authorization.oauth2",
    ...
    "claims_supported": [
        "address",
        ...
     ],
    "id_token_signing_alg_values_supported": [
        "none",
        ...
     ],
    "issuer": "https://localhost:9031",
    "jwks_uri": "https://localhost:9031/pf/JWKS",
    "ping_end_session_endpoint": "https://localhost:9031/idp/startSLO.ping",
    "ping_revoked_sris_endpoint": "https://localhost:9031/pf-ws/rest/
sessionMgmt/revokedSris",
    ...
    "revocation_endpoint": "https://localhost:9031/as/revoke_token.oauth2",
    ...
    "token_endpoint": "https://localhost:9031/as/token.oauth2",
    ...
    "userinfo_endpoint": "https://localhost:9031/idp/userinfo.openid",
    ...
}
```

The UserInfo endpoint (`userinfo_endpoint`) is where clients send access tokens to PingFederate in exchange for a list of additional claims about the users.

The JWKS (`jwks_uri`) endpoint returns a set of public keys that are generated and rolled automatically by PingFederate. Clients can use this information to verify the integrity of asymmetrically-signed ID Tokens.

For information about other parameters, see *OpenID Provider Metadata* in OpenID Connect Discovery 1.0 (openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata).

Web Service interfaces

PingFederate provides two built-in, SOAP-accessible Web Services related to Browser 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 114).
- 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 172.

PingFederate also provides the following REST-based Web Services:

- *OAuth Client Management Service* on page 471 — Useful for managing OAuth client applications, where needed (see *PingFederate OAuth AS* on page 60 and *Manage OAuth clients* on page 194).
- *OAuth Access Grant Management Service* on page 477 — Allows retrieval and revocation of access grants.
- *Session revocation API endpoint* on page 478 — 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 480 — Allows developers and non-developers to change PingFederate IdP Connections settings (see *Service provider SSO configuration* on page 308).

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:

```
https://<host_server>:<admin_console_port>/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).

Exporting manually

For information about using the administrative console to export SP connections at an IdP site, see *Via the Manage Connections screen* on page 246.

For information about exporting IdP connection at an SP site, see *From the Manage Connections screen* on page 319.

Using the connection service

The Connection Web Service exposes the following method for exporting connections:

```
public string getConnection( String entityId, String role,) throws
 IOException
```

Code sample

```
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 245 for SP connections or *Access IdP connections* on page 318 for IdP connections). You can also import the connections into PingFederate manually by copying them into the `<pf_install>/pingfederate/server/default/ data/connection-deployer` directory.

PingFederate scans this directory periodically and imports connections automatically.

Caution: Manually importing a connection always overwrites an existing connection with the same ID (the Web Service provides a switch to disallow this behavior, if desired—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

Below is example client code using the Apache AXIS libraries that invokes 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

Below is example client code using the Apache AXIS libraries that invokes 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:

`https://<host_server>:<admin_console_port>/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:

```
http(s)://<pf_runtime_host>:<runtime_port>/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 470).

- `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 172). 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. The service is provided primarily for organizations with many OAuth clients, allowing programmatic management of OAuth clients, as an alternative to using the administrative console (see *Manage OAuth clients* on page 194).

Note: This Web Service is active only if client records are managed in a database, rather than in XML files (the installed default storage method). For information on setting up database management for OAuth clients, see *Define an OAuth client data store* on page 154.

Tip: PingFederate administrative API can also be used to manage OAuth clients programmatically regardless if the client records are managed in XML files or in a database. For more information, see *PingFederate administrative API* on page 480.

Endpoints

The REST resources in the following sections are URL path extensions of the PingFederate runtime endpoint:

`http(s)://<pf_runtime_host>:<runtime_port>/pf-ws/rest`

For example: `https://my_corp.com:9031/pf-ws/rest/oauth/clients`

Note: `POST` and `PUT` methods described in this section require parameter name/value pairs formatted in JavaScript Object Notation (JSON).

Important: Applications must authenticate to the Web Service using HTTP Basic credentials specified in a PingFederate Password Credential Validator (see *Manage password credential validators* on page 173). The configured Credential Validator, in turn, must be selected in the OAuth AS configuration (see *Configure the AS settings* on page 184).

/oauth/clients

This resource accepts the methods *POST* on page 472, *PUT* on page 474 and *GET* on page 475.

POST

Description

Creates a new client based on the parameters provided in the request. Parameters correspond to administrative-console fields; for additional information, see *Manage OAuth clients* on page 194.

The required MIME type is `application/json`.

JSON Parameters

Parameter	Description
clientId	(Required) A unique ID for the client.
name	(Required) A descriptive name for the client.
refreshRolling	Indicates whether a new refresh token is issued with each new access token (see *Field descriptions*). Allowed values: `true` or `false`. When not provided, the global setting for the AS is used (see *Configure the AS settings* on page 184).
redirectUris	The URI(s) to which the OAuth AS redirects the resource owner's user agent after authorization is obtained. *Required* for Implicit and Authorization Code grant types (see `grantTypes` below).
logoUrl	A URL for the logo presented to the user on the grant revocation page.
secret	Client password or phrase, *required* for the Client Credentials grant type unless client-certificate authentication is needed (see next two parameters).
clientCertIssuerDn	Client TLS certificate issuer, *required* for the Client Credentials grant type unless a `secret` is provided.
clientCertSubjectDn	Client TLS certificate subject, *required* for the Client Credentials grant type unless a `secret` is provided.
description	A description of what the client application does, displayed in browser when the user is prompted for authorization.
persistentGrantExpirationType	Indicates whether to override the global setting for the AS (see *Configure the AS settings* on page 184). Allowed values: • `SERVER_DEFAULT` (the default) – Use the global setting for the AS. • `NONE` – Grants do not expire, regardless of the global setting. • `OVERRIDE_SERVER_DEFAULT` – Use with both of the `persistentGrant`* parameters below to set the expiration time period.
persistentGrantExpirationTime	An integer representing units of time for storage of persistent grants for this client—use with `persistentGrantExpirationTimeUnit` (see below) and only when `persistentGrantExpirationType` is set to `OVERRIDE_SERVER_DEFAULT`.

Parameter	Description
persistentGrantExpirationTimeUnit	Unit for the expiration time set in the parameter above (if applicable). Allow values: • `h` (hours) • `d` (days)
bypassApprovalPage	If set to `true`, user consent to resource access is assumed and the approval page is not presented.
restrictScopes	If set to `true`, limits client access to a subset of the scopes defined for the AS (see *Configure the AS settings* on page 184). Scopes are limited to the default scope and any listed for `restrictedScopes` (see below). If no scopes are listed and this parameter is `true`, only the default scope is available for the client.
restrictedScopes	When used with `restrictScopes`, limits access to the scope(s) provided in the JSON list, in addition to the default scope.
grantTypes	One or more grant types allowed for the client (see *Grant types* on page 61). Allowed values: • `authorization_code` • `password`(Resource Owner Password Credentials) • `refresh_token` (Use with `authorization_code` or `password`.) • `client_credentials` • `implicit` • `extension` (SAML 2.0 Bearer) • `urn:pingidentity.com:oauth2:grant_type:validate_ bearer` (Access Token Validation) **Note:** Provide at least one grant type. To provide multiple grant types, include them into a comma-separated list.
defaultAccessTokenManagerId	The default access token manager for this client (see *OAuth access token management* on page 186).
idTokenSigningAlgorithm	The JSON Web Signature (JWS) algorithm required for the ID Token. Allowed values: • `none` - No signing algorithm • `HS256` - HMAC using SHA-256 • `HS384` - HMAC using SHA-384 • `HS512` - HMAC using SHA-512 • `RS256` - RSA using SHA-256 • `RS384`- RSA using SHA-384 • `RS512` - RSA using SHA-512 • `ES256` - ECDSA using P256 Curve and SHA-256 • `ES384` - ECDSA using P384 Curve and SHA-384 • `ES512` - ECDSA using P521 Curve and SHA-512
policyGroupId	The desired Open ID Connect policy.
grantAccessSessionRevocationApi	If set to `true`, this client is allowed to access the Session Revocation API (see *Back-channel session revocation* on page 183).
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 (see *Asynchronous front-channel logout* on page 183).

Parameter	Description
logoutUris	A list of client logout URI's which will be invoked when a user logs out through one of PingFederate's SLO endpoints (see *Asynchronous front-channel logout* on page 183). Similar to `redirectedUris`, multiple entries are allowed.

Example JSON

```
{"client":[
  {
  "clientId":"12345",
  "name":"Client Doe",
  "refreshRolling":"true",
  "redirectUris":[
    "http://www.url.com",
    "http://www.url2.com"
    ]
  "logoUrl":"http://www.url.com/image.gif",
  "clientCertIssuerDn":"CN=CA, dc=pingidentity, dc=com",
  "clientCertSubjectDn":"cn=MyClient, dc=pingidentity, dc=com",
  "description":"Description of the client",
  "persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT",
  "persistentGrantExpirationTime":"3",
  "persistentGrantExpirationTimeUnit":"d",
  "bypassApprovalPage":"true",
  "restrictScopes":"true",
  "restrictedScopes":[
    "scope 1",
    "scope 2"
    ]
  "grantTypes":[
    "password",
    "refresh_token"
    ]
  }
]}
```

Returns

- 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

Description

Updates client details for a specified client.

Note: You cannot update a client ID—you must delete the client record and create a new one.

JSON Parameters

The same parameters described for *POST* on page 472 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.

Note: If the secret parameter is used without `forceSecretChange`, the secret value is ignored.

Example JSON

```
{"client":[
{
  "clientId":"12345",
  "name":"Client Doe",
  "refreshRolling":"true",
  "redirectUris":[
    "http://www.url.com",
    "http://www.url2.com"
    ]
  "logoUrl":"http://www.url.com/image.gif",
  "forceSecretChange":"true",
  "secret":"mysecretphrase",
  "description":"Description of the client",
  "persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT",
  "persistentGrantExpirationTime":"3",
  "persistentGrantExpirationTimeUnit":"d",
  "bypassApprovalPage":"true",
  "restrictScopes":"true",
  "restrictedScopes":[
    "scope 1",
    "scope 2"
    ]
  "grantTypes":[
    "password",
    "refresh_token"
    ]
  }
]}
```

Returns

- 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

Description

Retrieves details for all OAuth clients.

JSON Parameters

None.

Returns

- 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.

/oauth/clients/id

This resource accepts the methods GET and DELETE.

GET

Description

Retrieves details for the specified client ID.

JSON Parameters

None.

Returns

- 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

Description

Deletes records for the specified client ID.

Returns

- 200 – Success
- 400 – Failed To Delete Client

  The response contains details as to why client could not be deleted.
- 401 – Invalid Credentials

  The user does not exist or is not authorized.
- 405 – Method Not Allowed

  The client ID was not specified.
- 500 – Internal Server Error

  An unknown error has occurred.

Logging

PingFederate records the actions performed via this endpoint in the `runtime-api.log` file. While the events themselves are not configurable, you may adjust the `log4j2.xml` configuration settings to alter the format and desired level of detail surrounding each event.

Each log entry contains information relating to the event, including:

- Time the event occurred on the PingFederate server
- Administrator username performing the action
- Authentication method
- Client IP
- HTTP method
- REST endpoint
- HTTP status code

Each of the above fields is separated by a vertical pipe (|) for ease of parsing.

OAuth Access Grant Management Service

PingFederate includes a REST-based Web Service for OAuth access grant management. This service enables retrieval and revocation of access grants for a particular client or user.

The REST resources are URL path extensions of the PingFederate runtime endpoint:

`http(s)://<pf_runtime_host>:<runtime_port>/pf-ws/rest`

For example: `https://my_corp.com:9031/pf-ws/rest/oauth/clients`

Important: Applications must authenticate to the Web Service using HTTP Basic credentials specified in a PingFederate Password Credential Validator (see *Manage password credential validators* on page 173). The configured Credential Validator, in turn, must be selected in the OAuth AS configuration (see *Configure the AS settings* on page 184).

Endpoints: /oauth/clients/{clientId}/grants[/{grantId}] and /oauth/users/{userKey}/grants[/{grantId}]

These resources accept the methods GET which supports retrieval and DELETE which supports revocation of access grants.

Note: The Clients endpoint requires client records to be managed in a database, rather than in XML files (the installed default storage method). For information on setting up database management for OAuth clients, see *Define an OAuth client data store* on page 154.

Parameters

clientId	The client identifier to retrieve or revoke grants for (see *Configure an OAuth client* on page 194).
userKey	The user key to retrieve or revoke grants for. The user key is defined under IdP Adapter Mapping and Resource Owner Credentials Mapping.
grantId (optional)	Access grant identifier used to retrieve or revoke a specific grant. The value corresponds to the id field of the JSON array of grants returned from a previous `GET /oauth/clients/{clientId}/grants` or `GET /oauth/users/{userKey}/grants`. 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 request and JSON response

Request to retrieve all grants for client `im_client`:

```
GET /pf-ws/rest/oauth/clients/im_client/grants HTTP/1.1
Host: localhost:9031
Authorization: Basic YWRtaW46MkZlZGVyYXRl
X-XSRF-HEADER: PingFederate
```

Response could be:

```
{
    "items": [{
        "id":"gn3T3qGVjoHL9p8HKtCSZNriwg9H3DA8",
        "userKey":"joe",
        "grantType":"IMPLICIT",
        "scopes":[],
        "clientId":"im_client",
        "issued":"2014-03-26T21:15:38.551Z",
        "updated":"2014-03-26T21:15:38.551Z"
    }]
}
```

Returns

- 200 – Success
- 204 – Success with no content returned

  Will be seen when revoking an access grant
- 401 – Invalid Credentials.

  The user does not exist or the password is incorrect.
- 404 – Not found

  Resource (user, client, or access grant) 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.

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.

Important: OAuth clients must authenticate to the Web Service using their client secrets via HTTP Basic authentication or client certificates.

Endpoint: /pf-ws/rest/sessionMgmt/revokedSris

Tip: Information about the Session Revocation API endpoint is also available in the OpenID Connect provider metadata endpoint. Look for `ping_revoked_sris_endpoint` in the metadata.

This resource accepts the HTTP 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).

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 POST request and responses

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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT25tbGl
X-XSRF-HEADER: PingFederate
Content-Type: application/json

{"id":"abc123"}
```

Possible HTTP response status 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 GET request and responses

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
 YWNfb2ljX2NsaWVudDphYmMxMjNERUZnaGlqa2xtbm9wNDU2N3JzdHV2d3h5elpZWFdVVDg5MTBTUlFQT25tbGl
```

```
X-XSRF-HEADER: PingFederate
```

Possible HTTP response status 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.
- 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 // new
- 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 484. 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 role(s) 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 108.

3. Click **Server Configuration** on the Main Menu.
4. Click **Account Management** under Administrative Functions.
5. 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 481.

   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 481.

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 481.)

   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 160).

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 481.

   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 on the `pf.admin.https.port` defined 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/POST requests.

To access the interactive documentation in PingFederate:

1. Start PingFederate.
2. Launch your browser and go to:

   `https://<DNS_NAME>:9999/pf-admin-api/api-docs/`

   where `<DNS_NAME>` is the fully qualified name of the machine running the PingFederate server.

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: If you are upgrading to PingFederate 5.1 or higher and importing a configuration archive that uses expression mapping, the feature is enabled automatically.

This appendix 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 expressions by editing a configuration file located in:

`<pf_install>/pingfederate/server/default/data/config-store/`

Important: If the current configuration contains expressions, disabling the feature causes errors during runtime processing.

To enable or disable expressions:

1. In the directory cited above, open the file:

   `org.sourceid.common.ExpressionManager.xml`

   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:

   ```
   <item name="evaluateExpressions">true</item>
   ```

   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) Log on to the administrative console.
   b) Click **Server Configuration** on the Main Menu.

c) Click **Cluster Management** under Administrative Functions.
d) Click **Replicate Configuration**.
e) Restart PingFederate on the engine nodes.

When you enable expressions, these expressions are available for use in multiple locations:

- The drop-down menus under Source in each of the administrative-console contract fulfillment screens (the contract fulfillment screen, for example)
- The provisioning attribute-mapping screen (when outbound provisioning itself is enabled—see *Outbound provisioning for IdPs* on page 78)
- The **Show Advanced Criteria** section on the **Issuance Criteria** screen following each of the administrative-console contract fulfillment screens

When you make this selection, you can enter the expression in the text field provided. You can also test expressions (see *Using the OGNL edit screen* on page 488).

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
```

For more information, see *Using the OGNL edit screen* on page 488.

For more information about OGNL, including detailed user documentation, see the Apache Commons OGNL page (`commons.apache.org/ognl`).

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()
```

The returned value will be an instance of `javax.servlet.http.HttpServletRequest` (see *https://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 the expression below, 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"
```

Token Authorization

The expression below verifies if 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
```

Note: Line breaks are inserted for readability only; statements calling methods whose arguments are enclosed in quotes must be entered on a single line.

The following expression extracts the domain information out of an email address (mail) and returns true if it matches a specific domain (`company.com`):

```
#this.get("mail")!=null?
(
  #email=#this.get("mail").toString(),
  #atSign="@",
  #at=#mail.indexOf(#atSign),
  #at > 0?
    (
      #domain=#mail.subject(#at+1),
      #domain.matches("(?i)company.com")
    ):false
):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.

The following 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::74
```

The `isInRange` method supports both IPv4 and IPv6 CIDR notations.

HTTP Request Context

The example below may be used to retrieve a value from an HTTP request object, in this case the `User-Agent` HTTP header, for comparison to a value required for token authorization.

```
#this.get("context.HttpRequest").getObjectValue().getHeader("User-
Agent").equals("somevalue")
```

STS Client Authentication Context

The STS SSL Client Certificate Chain example below 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"))
```

In the example above,

`#this.get("context.StsSSLClientCertChain").getObjectValue()`

returns an array of `java.security.cert.X509Certificate` instances (see *https://docs.oracle.com/javase/8/docs/api/java/security/cert/X509Certificate.html*). This array starts with the client certificate itself.

Issuance criteria and multiple virtual server IDs

When you use virtual server IDs to connect to multiple environments in one connection (see *Connecting to a partner in multiple connections* on page 81), 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.

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.

Note: For information about using OGNL, refer to the *Apache Commons OGNL page* (`commons.apache.org/ognl/index.html`).

- 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.

  The **Edit** action is also available on Issuance Criteria screens for expressions (see *About token authorization* on page 71).

  Note: The test function does not work for the `context.httpRequest` attribute, because it's value is an object rather then text.

  Here is an example of the edit screen, from the IdP configuration flow:

To test an expression:

1. Enter an input value in the Value text box associated with the attribute.
2. Click the **Test** link near the bottom right of the 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 already done so, enable OGNL expression by editing the `org.sourceid.common.ExpressionManager.xml` file 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

Message Types	Available Variables Classes/Interfaces in Javadoc
	#AssertionTypes org.sourceid.saml20.xmlbinding.assertion.AssertionType[] #Attributes org.sourceid.util.log.AttributeMap
ResponseDocument	#ResponseDocument org.sourceid.saml20.xmlbinding.protocol.ResponseDocument #Attributes org.sourceid.util.log.AttributeMap

SP Connections (SAML 1.x)

Message Types	Available Variables Classes/Interfaces in Javadoc
AssertionType	#AssertionType org.sourceid.protocol.saml11.xml.AssertionType #AssertionTypes org.sourceid.protocol.saml11.xml.AssertionType[] #Attributes org.sourceid.util.log.AttributeMap
ResponseDocument	`#ResponseDocument` org.sourceid.protocol.samlp11.xml.ResponseDocument #Attributes org.sourceid.util.log.AttributeMap

SP Connections (WS-Federation)

Message Types	Available Variables Classes/Interfaces in Javadoc
AssertionType	#AssertionType org.sourceid.protocol.saml11.xml.AssertionType #Attributes org.sourceid.util.log.AttributeMap
RequestSecurityToken ResponseDocument	#RequestSecurityTokenResponseDocument org.xmlsoap.schemas.ws.x2005.x02.trust.RequestSecurityTokenResponseDocument #Attributes org.sourceid.util.log.AttributeMap

IdP Connections (SAML 2.0)

Message Type	Available Variables Classes/Interfaces in Javadoc
AuthnRequestDocument	#AuthnRequestDocument org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument

Other Available Variables (regardless of roles and protocols)

Available Variables	Classes/Interfaces in Javadoc
#XmlHelper	com.pingidentity.sdk.xml.XmlHelper
#HttpServletRequest	javax.servlet.http.HttpServletRequest
#HttpServletResponse	javax.servlet.http.HttpServletResponse

Variables related to Federation Hub (regardless of message type)

Connections	Protocol	`Available Variables` Classes/Interfaces in Javadoc
SP and IdP connections	SAML 2.0	#FedHubIncomingAuthnRequest org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument
SP connection	SAML 2.0	#FedHubOutgoingAuthnRequest org.sourceid.saml20.xmlbinding.protocol.AuthnRequestDocument
SP connection	SAML 2.0	#FedHubIncomingAuthnResponse
	SAML 1.x	org.sourceid.saml20.xmlbinding.protocol.ResponseDocument (SAML 20)
	WS-Federation	org.sourceid.protocol.samlp11.xml.ResponseDocument (SAML 1.x) org.xmlsoap.schemas.ws.x2005.x02.trust. RequestSecurityTokenResponseDocument (WS-Federation)
SP connection	SAML 2.0	#FedHubIdpConnPartnerId
	SAML 1.x	java.lang.String
	WS-Federation	The **Partner's Entity ID** in the IdP connection that bridges the identity provider.
SP connection	SAML 2.0	#FedHubIdpConnProtocol
	SAML 1.x	java.lang.String
	WS-Federation	The protocol of the SP connection. The returned values are SAML20, SAML11, SAML10, or WSFED.
IdP connection	SAML 2.0	#FedHubSpConnPartnerId
	SAML 1.x	java.lang.String
	WS-Federation	The **Partner's Entity ID** in the SP connection that bridges the service provider.
IdP connection	SAML 2.0	#FedHubSpConnProtocol
	SAML 1.x	java.lang.String

Connections	Protocol	`Available Variables` **Classes/Interfaces in Javadoc**
	WS-Federation	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 in this workflow.

Sample inputs (attributes and their values):

Attribute Name	ExtAttr1
Attribute Name Format	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified
Attribute Value	123

Attribute Name	ExtAttr2
Attribute Name Format	urn:pingidentity.com:SAML:attrname-format:xml:complex
Attribute Value	<saml:Attribute xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" Name="ExtAttr2" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"> <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:customNs="http://www.sample.tld/customnamespace"> <customNs:Line>Documentation</customNs:Line> <customNs:Line>Ping Identity</customNs:Line> </saml:AttributeValue> </saml:Attribute>

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>/sp/startSSO.ping?PartnerIdpId=<IdP>&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

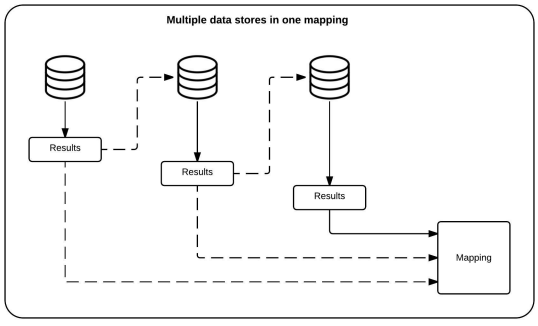
PingFederate can be configured to use data stores to supply attributes for various contracts (for examples, the IdP's adapter contract, IdP's attribute contract, the SP's adapter contract, or STS token contracts) or conditions to be verified in the token authorization process. Standard data stores may include any JDBC-accessible database or an LDAP v3-compliant directory server. 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*.

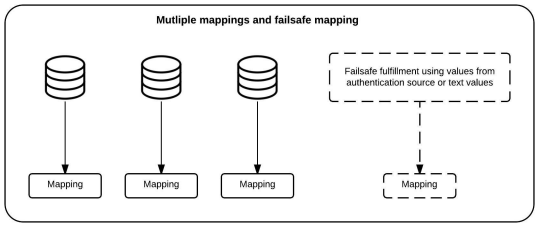
If a required attribute (such as `SAML_SUBJECT` in a SAML contact or `subject` in an authentication policy contract) cannot be fulfilled, the request fails.

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.



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 workflow using the results from the data store queries.

Choose a data store

From the **Data Store** screen, choose a data store for PingFederate to look up attributes.

1. Enter an ID and a description 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 496 • *Enter a database search filter* on page 496
LDAP	• *Specify an LDAP directory and attributes* on page 497 • *Define encoding for LDAP binary attributes* on page 498 (optional) • *Enter an LDAP search filter* on page 499

Data store type	Required tasks
Custom	• *Specify a custom source filter and fields* on page 499

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 *http://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**.

   Repeat as needed to add more columns.

   Note: It is not required to add a column here for it to be used as part of a search filter. Add only the attributes required for contract fulfillment or token authorization.

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 in 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 LDAP 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 class containing the desired attributes.
Attributes	A list of attributes based on the selected **Root Object Class** value. Subject DN, which may be used as the primary user identifier, is always returned by default.

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**.

   It is not required to add an attribute here for it to be used as part of a search filter. Add only the attributes required for contract fulfillment or token authorization.

   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)

   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 uses objectGUID, an immutable Active Directory binary attribute associated with user accounts 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` **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 in the ***the applicable SP connection*** > **Browser SSO** > **Assertion Creation** > **Authentication Source Mapping** > ***the applicable authentication source*** > **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

You can review the configuration on the **Summary** screen.

To change your configuration, click the heading over the information you want to edit.

When you are finished, click **Done** to continue with the rest of the configuration, where you can use the attribute values returned from the data store in the applicable contract fulfillment screen, the issuance criteria screen, or both, to fulfill your use case.

Note: Remember to click **Save** at the end of your workflow to save your settings.

Troubleshooting

Basic troubleshooting tips are provided here to help overcome common difficulties. Help is also available from the *Support Center* at *pingidentity.com*.

This appendix contains the following sections:

- *Data store issues* on page 501
- *Installation issues*
- *Other runtime issues* on page 505
- *Server startup* on page 501

Server startup

Problem	Solution
PingFederate does not start.	Make sure that Oracle Java SE Runtime Environment (Server JRE) is installed (see *Install Java* on page 17 in the "Installation" chapter of the *Get started with PingFederate* guide).
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 88).

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 144.
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 145). 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 160).

Troubleshooting SSO or SLO failures

There are three common types of SSO or SLO failures:

- *Resolve URL-related errors* on page 501
- *Resolve service-related errors* on page 502
- *Troubleshoot runtime errors* on page 502

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-initated 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 transaction 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 transaction ID in templates

You can configure PingFederate 7.2 (or higher) to display the transaction ID in the user-facing error Velocity templates. When an error occurs, you can use the transaction ID to look for the related log messages. The Velocity variable is `$TrackingId` and is available in the following templates:

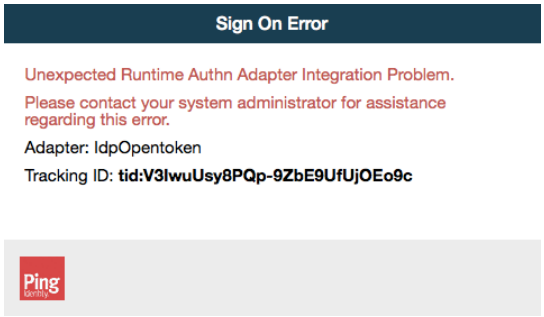
- `general.error.page.template.html`
- `generic.error.msg.page.template.html`
- `idp.slo.error.page.template.html`
- `idp.sso.error.page.template.html`
- `sourceid-wsfed-idp-exception-template.html`
- `sp.slo.error.page.template.html`
- `sp.sso.error.page.template.html`
- `state.not.found.error.page.template.html`

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 transaction ID.



Correlate log messages by transaction ID

All server log messages (except the contents of the inbound requests and the outbound responses) are prefixed with their respective transaction 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 transaction ID in the server log, for example:
3. Use the transaction 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 transaction ID was `V3IwuUsy8PQp-9ZbE9UfUjOEo9c`. Based on the transaction 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 transaction 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 transaction IDs, use the transaction 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 transaction 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.

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 130).
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—used 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 64).

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; i.e., 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 web site user, attributes may include a name, group affiliation, email address, etc.

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.

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 web site 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.

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

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

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 Authorization Server

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 OAuth Authorization Server to obtain access tokens to do so.

Online Certificate Status Protocol

(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.

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*.

policy

A set of rules for handling security token requests in PingFederate.

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

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.

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>

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

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

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

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

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 web site (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

Identifies a web resource with a string of characters conforming to a specified format.

Uniform Resource Locator

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

A standard mechanism for securing Web Service interactions, often by binding a security token to the Web Service request.

Web Services

Nonbrowser-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

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

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

Ping Identity®'s 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* on page 515
- *Deploy cluster servers* on page 516
- *Deploy provisioning failover* on page 528
- *Configuration synchronization* on page 529

Overview of clustering

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.

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 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 *Guide*.

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

PingFederate abstracts its runtime session-state management 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 (see *Runtime state-management services* on page 523).

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 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. 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 *Deployment architecture* on page 519).

The PingFederate software distribution does not contain a load balancer. Numerous hardware or software products are available commercially, including free downloads.

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 (see the next section).

Note: The PingFederate administrative console node must be configured to run outside of the load-balanced group to successfully process SSO requests.

Deploy cluster servers

Follow the steps below 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 this section and refer to information under *Deploy provisioning failover*.

1. For each node in a cluster, install PingFederate.

   Tip: You need only install one license file on any one machine in the cluster; the key will be pushed out to the other servers.
2. For each server instance, edit clustering properties in the `run.properties` file located in the `<pf_install>/pingfederate/bin` directory (see *Configure clustering properties* on page 516).
3. Optional: Adjust configuration files that control cluster architecture and runtime session-state management (see *Deployment architecture* on page 519 and *Runtime state-management services* on page 523).
4. Optional: If Outbound Provisioning is configured at your site and you want to provide failover capabilities, identify provisioning failover nodes (see *Deploy provisioning failover*).
5. Start or restart all of the PingFederate servers.

   Important: Start the server containing the license file before starting the other servers.
6. After you configure (or reconfigure) PingFederate through the administrative console, replicate the configuration on all nodes in the cluster (see *Console configuration push*).

Configure clustering properties

The clustering configuration options are maintained in the `run.properties` file located in the `<pf_install>/pingfederate/bin` directory. Configure the clustering properties on each node in the cluster. For information about each property, refer to the following table:

Property	Description
pf.operational.mode	Controls the operational mode of the PingFederate server. Refer to the properties file for a list of valid values and descriptions. Note that the value `STANDALONE` should not be used in a cluster unless user-state management is not needed for any reason and configuration-archive deployment is used as the configuration synchronization method (see *Configuration synchronization*).
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`.)

Property	Description
	If no value is set for the node index, the system assigns a default index derived from the last two octets of the IP address. We recommend, however, that you assign static indexes. **Tip:** The PingFederate Cluster Node Selector, available in version 7.0 and higher, provides the capability of overriding adapter processing based on the runtime node servicing a request.
pf.cluster.auth.pwd	Sets the password that each node in the cluster must use to authenticate when joining the group. 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.
pf.cluster.encrypt	Indicates whether or not to encrypt network traffic sent between nodes in a cluster. (Values: `true` | `false`.) The default value is `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.
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 group communication should bind. For machines with more than one network interface, you can use this property to increase performance (particularly with UDP) as well as improve security by segmenting group-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`).
pf.cluster.bind.port	Specifies the port associated with the bind-address property above or with the default network interface used. 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 one of the cluster's failure-detection mechanisms. If zero or unspecified, a random available port is used. The default value is `7700`.
pf.cluster.transport.protocol	Indicates the transport protocol used for group communication. (Values: `udp` | `tcp`.) The default value is `tcp`. Use UDP multicast when IP multicasting is enabled in the network environment and the majority of group traffic is point-to-full-group. You must also provide values for the pf.cluster.mcast* properties described below. 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

Property	Description
	point-to-few messaging than point-to-group. You must also provide a value for the pf.cluster.tcp.discovery.inital.hosts property described below. **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. Administrators can customize either stack or add to it as needed by modifying the associated configuration file.
pf.cluster.mcast.group.address	Defines the IP address shared among nodes in the same cluster for UDP multicast communication; required when UDP is set as the transport protocol. (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 use the same address for this property and the port property below.
pf.cluster.mcast.group.port	Defines the port associated with the property above, pf.cluster.mcast.group.address.
pf.cluster.tcp.discovery.initial.hosts	Designates the initial hosts to be contacted for group membership information when discovering and joining the group; required when TCP is set as the transport protocol. The value is a comma-separated list of host names (or IP addresses) and ports. Example: `host1[7600],10.0.1.4[7600],host7[1033],10.0.9.45[2231]` Add at least one group member that is known in advance and statically configured to this property on each node (because TCP does not provide the built-in group semantics of IP multicast). In practice, it is recommended that as many hosts as possible be included for this property on each node, to increase the likelihood of new members finding and joining the group. Alternatively, leave this property blank and enable dynamic discovery in the `<pf_install>/pingfederate/server/default/conf/tcp.xml` file.
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. Do not change the property values.

Note: This table does not include information about properties used for provisioning failover (see *Deploy provisioning failover*).

1. Configure the clustering properties for one of the nodes in the cluster.
2. Repeat this process on the rest of the nodes in the cluster.

Important: You must manually configure the clustering properties in the `run.properties` file 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 servers must be restarted if running.

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. Dynamic discovery 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 both 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. You enter the storage location and the access credentials into the `tcp.xml` file on each node in the cluster. 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.
3. Restart PingFederate if it is running.
4. Repeat these steps on the rest of the nodes (and any new nodes, as needed).

Tip: After the initial setup, your nodes are ready to be deployed, undeployed, and redeployed as traffic volume changes.

Note: Dynamic discovery is the mechanism of how nodes find their cluster. It does not impose a particular deployment architecture. With planning and additional configurations, you can set up a PingFederate cluster that works for your environment (see *Deployment architecture* on page 519).

Deployment architecture

In a cluster consisting of a large number of nodes, it may be desirable to share data with only a subset of nodes (for performance reasons). To facilitate this, most of the group RPC-based service implementations (see *Runtime state-management services* on page 523) 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 data.

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 *Guide* for detailed information about each implementation.

Configuration File	RPC-Based Service Implementation
cluster-artifact.conf	*Artifact-message persistence and retrieval Service* on page 526
cluster-assertion-replay-prevention.conf	*Assertion replay prevention service* on page 525
cluster-idp-session-registry.conf	*IdP session registry service* on page 525
cluster-inter-request-state.conf	*Inter-request state-management (IRSM) service* on page 523
cluster-session-revocation.conf	*Back-channel session revocation service* on page 527
cluster-sp-session-registry.conf	*SP session registry service* on page 525

The following table describes properties contained in the configuration files (the Artifact-Message Persistence and Retrieval Service uses only `rpc.timeout`):

Property	Description
preferred.node.indices	A comma-separated list of indices identifying the nodes with which this server shares session-state data for the associated service. If blank, all nodes in the cluster are used.
rpc.timeout	How long, in milliseconds, the server waits before timing out unresponsive RPC invocations.
synchronous.retrieve.majority.only	Indicates how many responses to wait for when making synchronous RPC calls (values: `true` | `false`). When `true` (the default), the server waits for the majority of recipients to respond. When `false`, it waits for all recipients to respond.
bulk.revoked.sris.timeout (found only in cluster-session-revocation.conf)	A node downloads a full revocation list from another node during startup or when it rejoins a cluster after 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` milliseconds (10 seconds).
read.local.only (found only in cluster-session-revocation.conf)	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 `true` (the default), 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 527).

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.

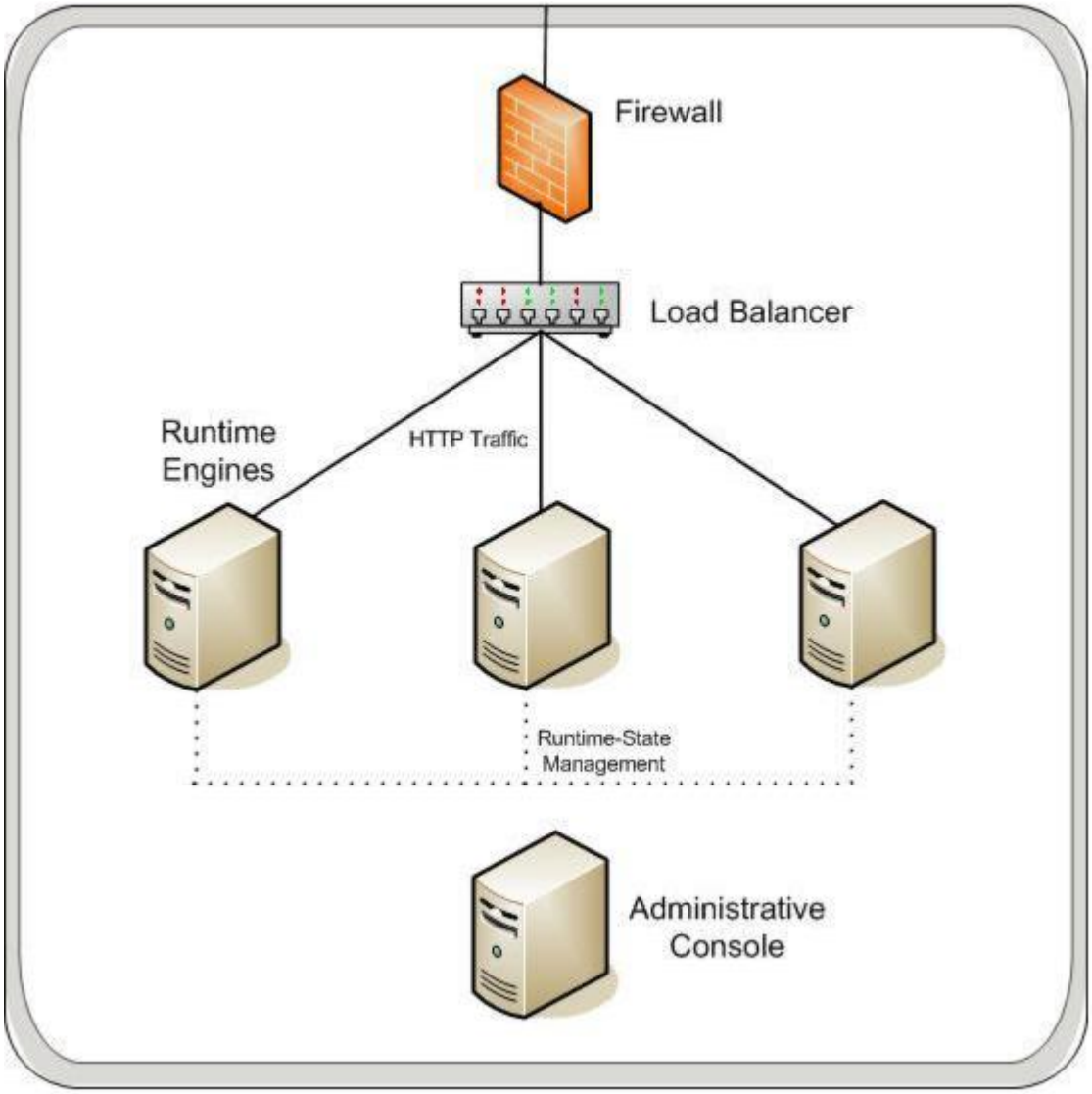
The use of preferred nodes can translate into any number of deployment configurations. Three primary strategies are discussed in the following sections and may serve as touch points for you to consider in conjunction with your network requirements:

- *Sharing all nodes* on page 520
- *Designating state servers* on page 521
- *Defining subclusters* on page 522

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

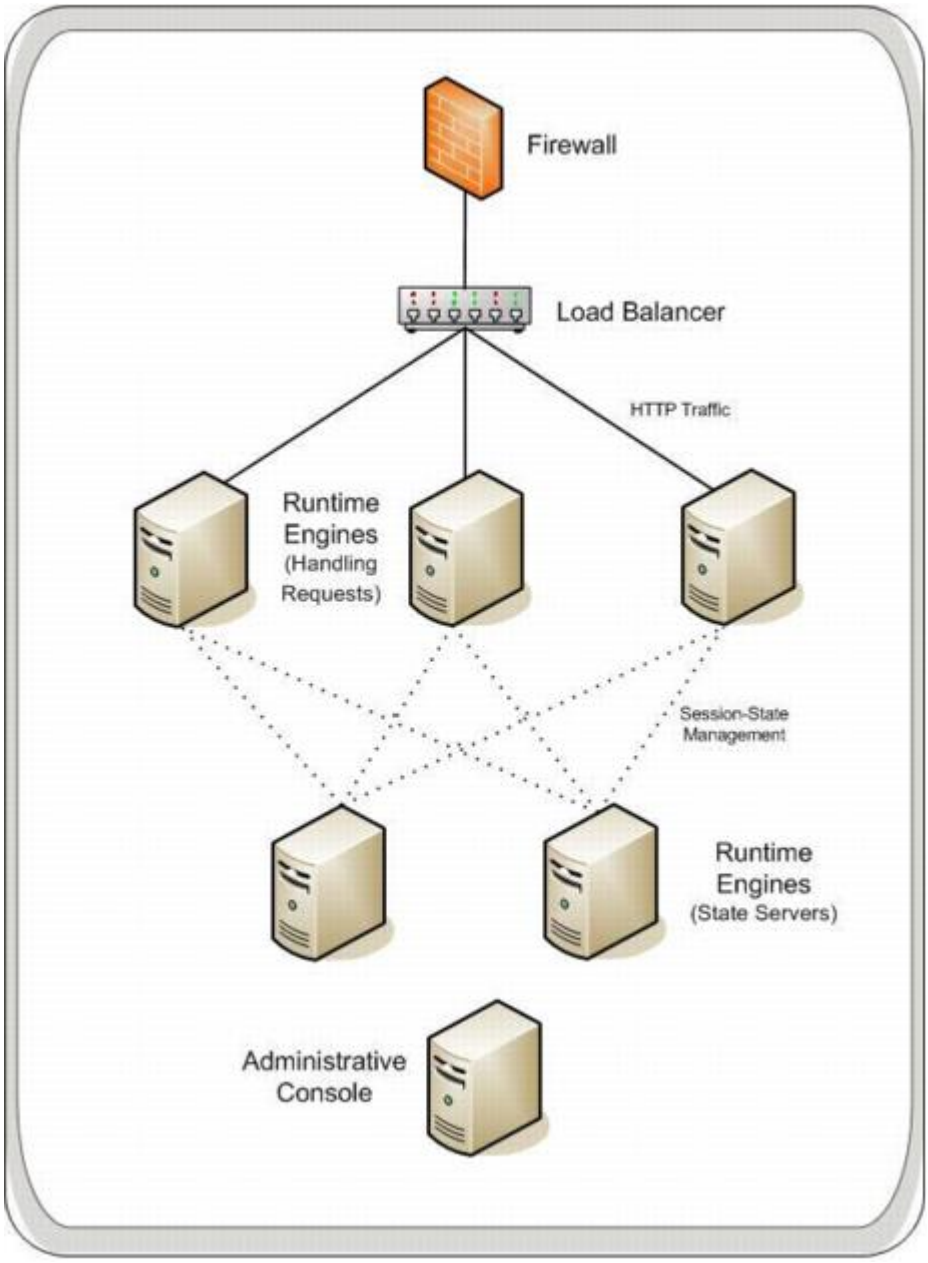
Leaving the preferred.node.indices property blank in all the cluster-configuration files provides a basic deployment case. An advantage of this approach is simplicity, including the option of using straightforward load-balancing strategies such as round robin. A disadvantage is that as additional nodes are added, the throughput improvement rate that clustering offers may decline as the state-replication overhead increases.



The diagram above illustrates the node-sharing approach. HTTP traffic is directed to all nodes.

Designating state servers

You can select a few 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. Load balancers 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 connections to every existing node; there need be only a connection between each server and each state server.

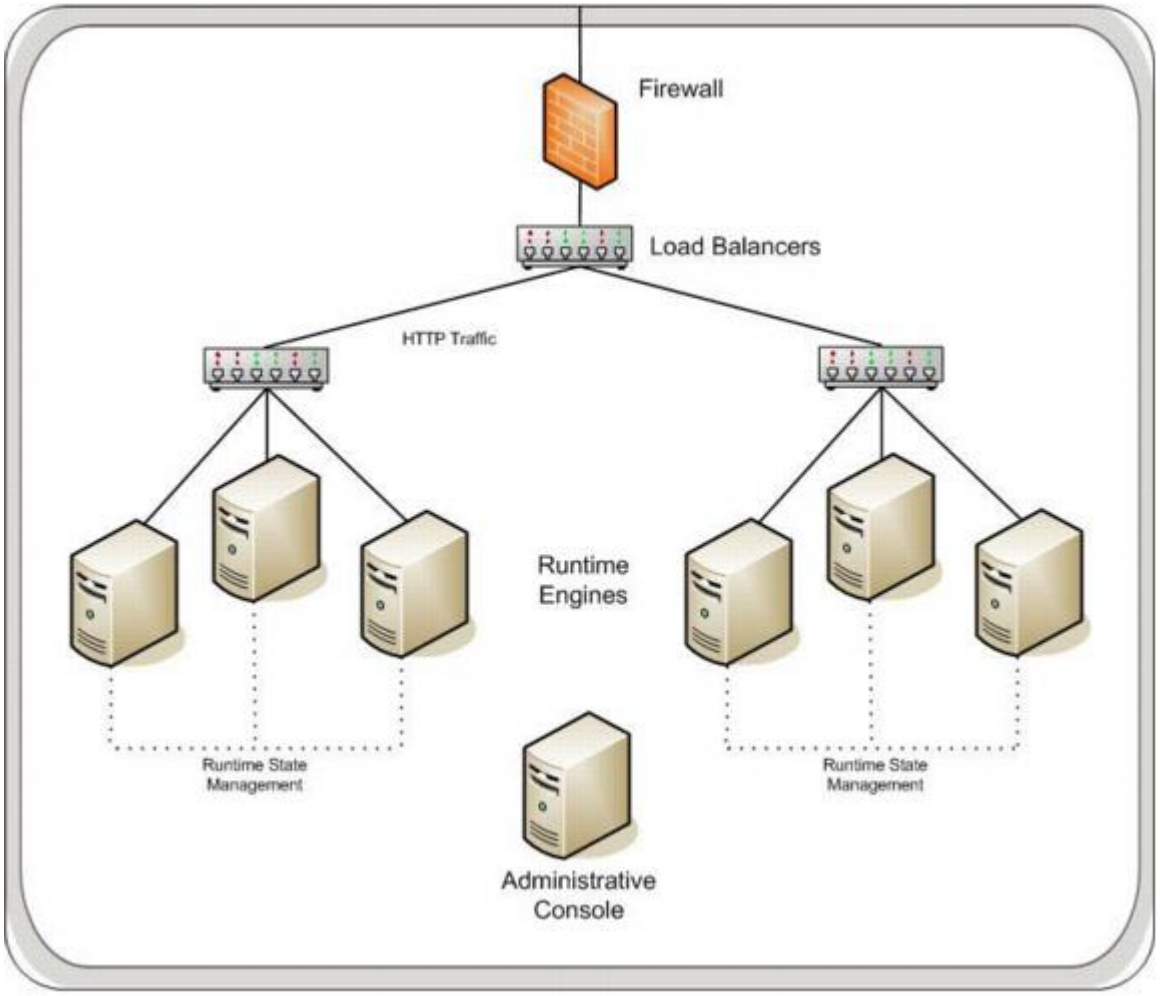


This diagram illustrates the state-server approach. If the two state-server nodes have indices of 1 and 2, then the preferred-nodes property of the other runtime nodes would be:

```
preferred.node.indices=1,2
```

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 data only with other members of the subcluster. This approach requires the load balancer to persist, or *stick*, user sessions to the same group so that each subsequent request from the same user is directed to one of the nodes in that group. (A wide variety of persistence mechanisms exist and may vary by load balancer – consult the documentation specific to your product.) 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.



The diagram above illustrates the subcluster approach. The preferred-nodes property of each server in the cluster lists the indices of all nodes in its subgroup (including itself). HTTP traffic is directed to all nodes, but the load balancer directs user sessions to the same subcluster.

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 implementations that make use of preferred nodes are in the directory `<pf_install>/pingfederate/server/default/conf` (see *Deployment architecture* on page 519).

Inter-request state-management (IRSM) service

The PingFederate server tracks user-session state information between HTTP requests – for example, when PingFederate, acting as an Identity Provider (IdP), redirects a user's browser to another system for authentication. When the user's browser returns to PingFederate after authentication, the server needs access to the state associated

with that user from before the 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 uses the preferred-nodes concept. `<pf_install>/pingfederate/server/default/conf/cluster-inter-request-state.conf` is the configuration file. All three preferred-node approaches described under *Deployment architecture* on page 519 are possible with this implementation. The subgroup approach, however, may be most appropriate.

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 *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 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
```

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
```

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.

All preferred-node approaches are possible with this implementation (see *Deployment architecture* on page 519). However, if your site accepts single logout messages via SOAP, the subgroup approach may not succeed because the load balancer's sticky-session functionality cannot ensure that the SOAP message from a partner will be directed to the appropriate subgroup.

The service proxy uses the class:

`org.sourceid.saml20.service.impl.grouprpc.IdpSessionRegistryGroupRpcImpl`

SP session registry service

PingFederate uses the SP (Service Provider) 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.

All preferred-node approaches are possible with this implementation (see *Deployment architecture* on page 519). However, if your site accepts single logout messages via SOAP, the subgroup approach may not succeed because the load balancer's sticky-session functionality cannot ensure that the SOAP message from a partner will be directed to the appropriate subgroup.

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 and the IdP and SP session registries 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 (Security Assertion Markup Language) standards specify that when an SP receives assertions via the POST binding, the SP should keep track of that assertion for the duration of its validity to ensure that it is not replayed (that is, intercepted by a third party and re-posted). PingFederate delegates that responsibility to the assertion replay prevention service. This service is used only when PingFederate is acting as an SP and receives assertions via the POST binding (see various binding profiles under *SAML 1.x profiles* and *SAML 2.0 profiles*.

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.

Due to the nature of the threat that this service is intended to prevent, only the *Sharing all nodes* on page 520 and *Designating state servers* on page 521 deployment strategies are appropriate for this service (see after *Deployment architecture* on page 519).

The service proxy uses the class:

`org.sourceid.saml20.service.impl.grouprpc.AssertionReplayPreventionServiceGroupRpcImpl`

This service is unique in that it fulfills only a security condition of the federation specifications, 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 use 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.saml2" index="1"/>
<ArtifactResolutionService Binding="..." Location="https://node2/idp/
ARS.saml2" index="2"/>
<ArtifactResolutionService Binding="..." Location="https://node3/idp/
ARS.saml2" index="3"/>
<ArtifactResolutionService Binding="..." Location="https://node4/idp/
ARS.saml2" 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 (see *Session revocation API*).

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 (see *Deployment architecture* on page 519).

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 (see *Configure OAuth AS settings*), 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`.

Extensibility and other services

Custom implementations of all state management services described in this *Guide* can be plugged into PingFederate as needed for special deployments. Software developers can refer to the Javadoc reference:

`<pf_install>/pingfederate/sdk/doc/index.html`

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 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 HyperSQL database cannot be used in a failover configuration. Each server in the failover network must be configured to use the same relational database (see *Configure outbound provisioning settings*).

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 (the default). • `STANDALONE` – Provisioning is enabled, without failover. • `FAILOVER` – Provisioning is enabled, with failover. **Important:** The `STANDALONE` setting is not used for failover configuration; primary and secondary servers must be set to `FAILOVER`.
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.

Property	Description
	Important: The active primary server *must* have an index number of `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 **Synchronization Frequency** set in the **Server Configuration** > **Server Settings** > **Outbound Provisioning** screen on the administrative console.

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 update the **Internal Provisioning Data Store** setting in the **Server Configuration** > **Server Settings** > **Outbound Provisioning** screen (see *Configure outbound provisioning settings*).

   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

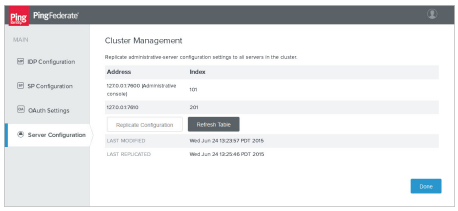
All nodes in a PingFederate clustered environment must have the same configuration settings, as set via the administrative console. You can use any of the following methods to ensure that configuration data is synchronized on all cluster nodes:

- 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. The Connection Management Service requires authentication. For more information, see *Web Service interfaces*.

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 to the administrative-console configuration, a message appears at top of the console directing the administrator to this screen.



From this screen, replicate the current console configuration to all the server nodes in the cluster if the **Last Modified** timestamp is later than the **Last Replicated** timestamp.

Tip: 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.

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 and then deploying it (either manually or via a scripted process) to each cluster node or provisioning-failover server. For more information about creating and deploying configuration archives, see *Manage configuration archives* .

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.

Note: You must still configure each node's session-state services, if you are using settings other than defaults (see *Deployment architecture*). The archive does not contain information about clustering; it contains only information about server settings and partner configuration.

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* on page 531
- *SSO integration concepts* on page 531
- *Identity provider integration* on page 532
- *Service provider integration* on page 534
- *Summary* on page 536

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 for download at *pingidentity.com* (`www.pingidentity.com/en/products/downloads.html`). User guides for the kits can be found under *Product Documentation* (`documentation.pingidentity.com`).

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 539* for more information, available in the PingFederate distribution `sdk` directory.

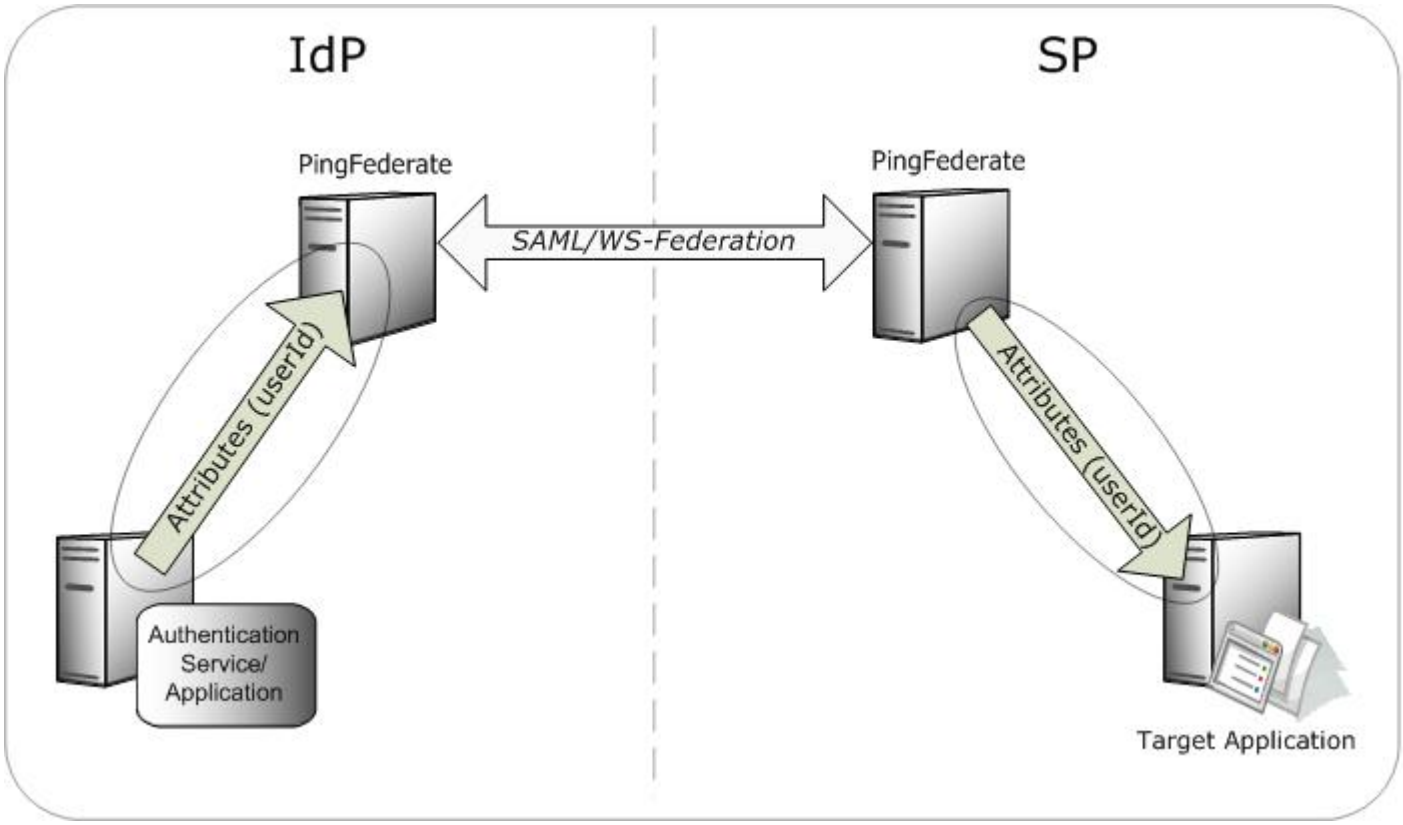
In addition, for integration with the PingFederate WS-Trust security token service (STS), we provide a range of *Token Translators*. These plug-in token processors (for an IdP) and token generators (for an SP) connect the STS with Web Service providers and clients for access to identity-enabled Web Services.

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 28 in the *Get started with PingFederate* guide.) 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.

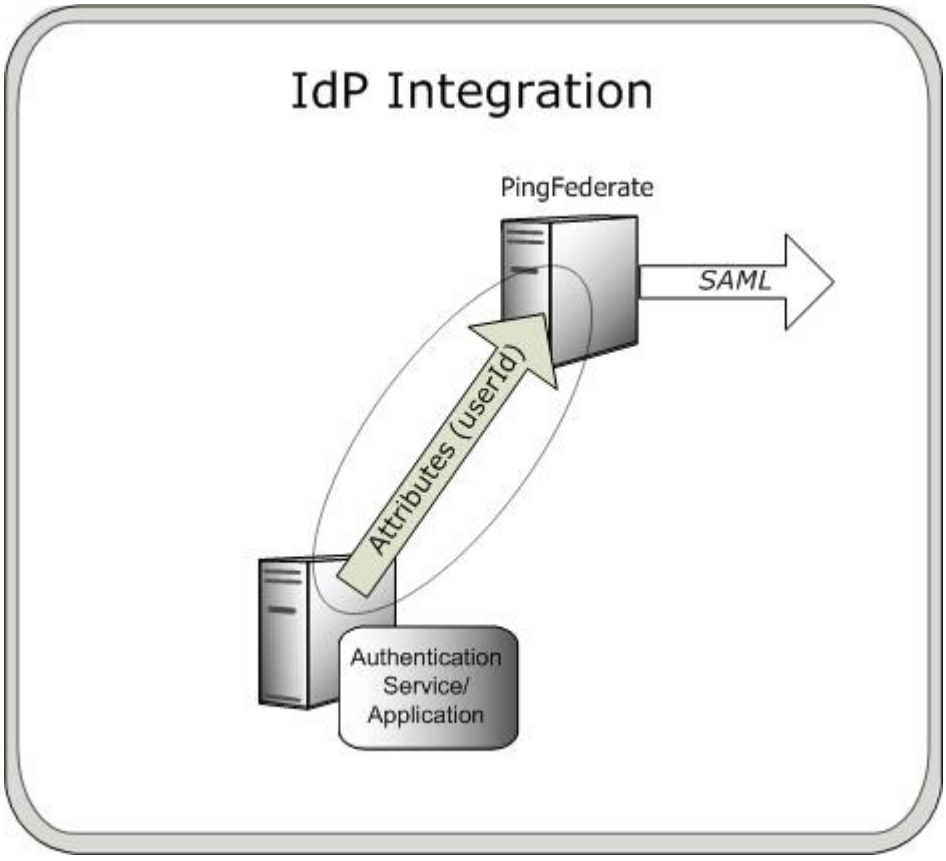
The following diagram illustrates the basic concepts of integration with PingFederate:



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 for automated user provisioning. See details under *Workforce to the Cloud* at the Ping Identity web site.



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.

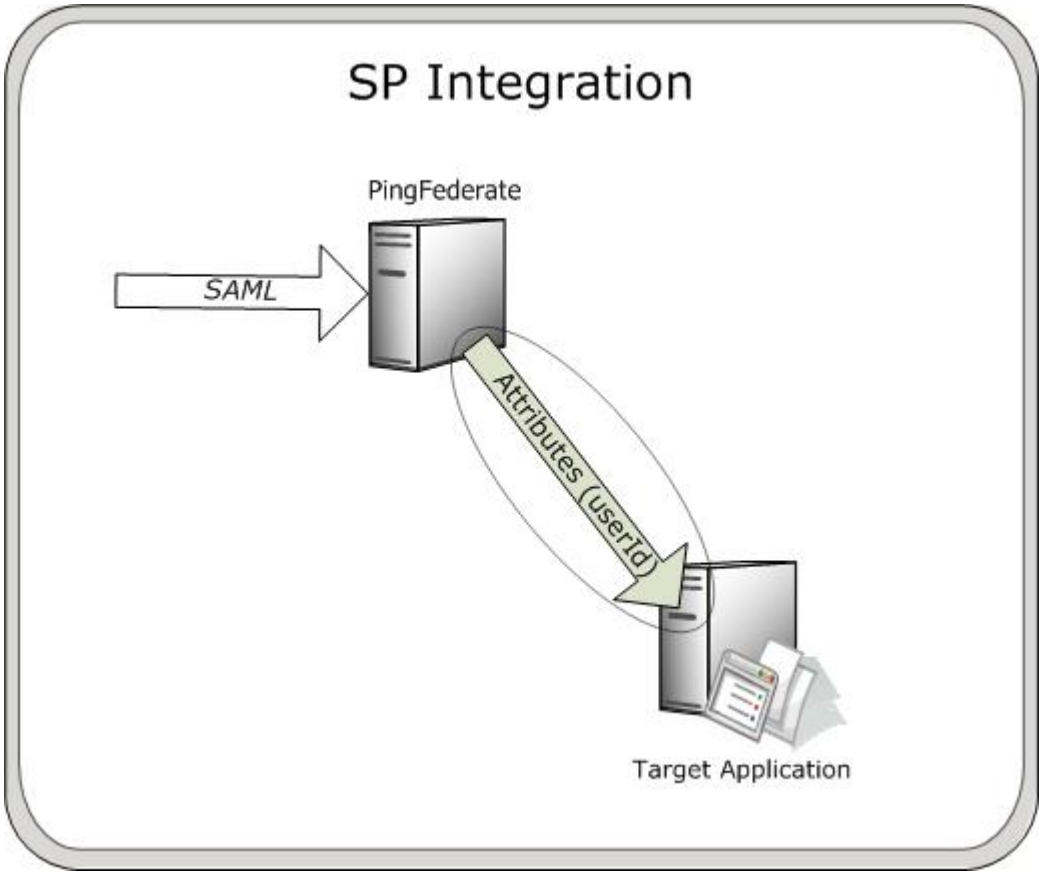
Authentication integration kits do not require any development; integration with PingFederate is accomplished entirely through the PingFederate administrative console. Ping Identity offers integration kits for authentication systems including:

- 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 including, 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	Agentless Integration Kit
	Java Integration Kit	Java Integration Kit
	PHP 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)	Not applicable
	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	
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 Conenctor
		Workday Connector
		Zendesk Connector

Ping Identity continues to develop new bundled adapters (included with PingFederate) and integration kits; check the Ping Identity *download site* (`www.pingidentity.com/en/products/downloads.html`) for the most up-to-date list of kits.

Please find *User Guides* and other documentation for current integration kits under *Product Documentation* on the web site (`documentation.pingidentity.com`).

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* on page 539
- *SDK introduction* on page 540
- *Get started with the SDK* on page 541
- *Implementation guidelines* on page 542
- *Build and deploy your project* on page 556

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 531*, described under *Additional Documentation* on page 539.)
- 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 531* 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 56* provides background information and user-interface (UI) configuration details needed to integrate implementation(s) of PingFederate interfaces.
- Integration Kit User Guides for *Java*, *.NET*, and *PHP* show examples of SDK implementations.

Related Publications

You may download related publications for offline viewing from *documentation.pingidentity.com*.

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 *Downloads* page (`www.pingidentity.com/en/products/downloads.html`) 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 selectors 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 LDAP or JDBC 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 Center* `(ping.force.com/Support)` 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 541
- *Set up your project* on page 541

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 556.

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 542
- *Describable plug-in* on page 542

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 `sp-adapter-example` 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 542, 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.

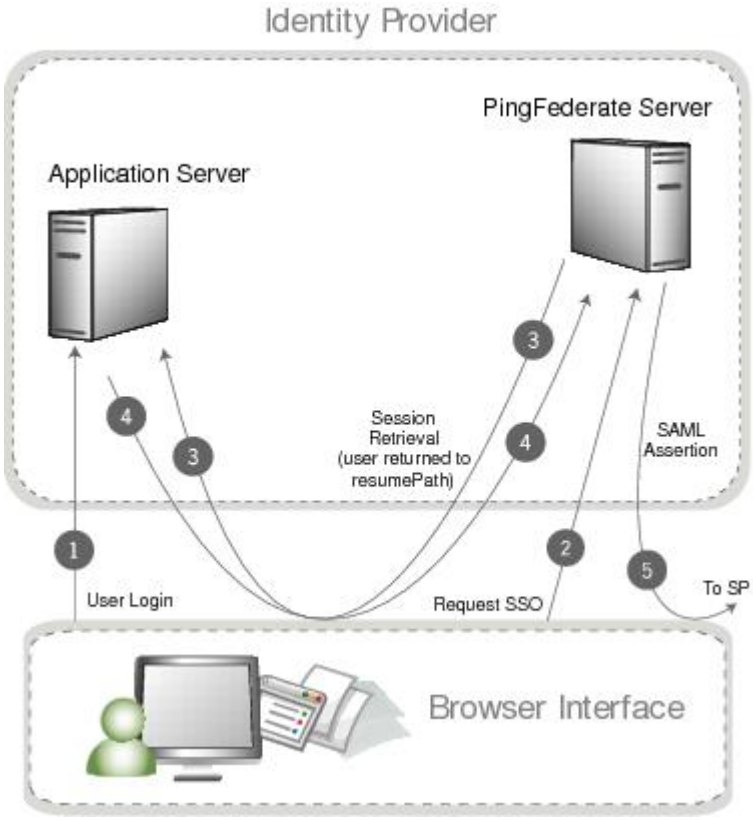
Tip: When creating a custom adapter, you can design it to render a template for processing and returning HTML to the user's browser using the `TemplateRendererUtil`. A sample `(template-render-adapter-example)` is included in the `sdk/plugin-src` directory of your PingFederate instance.

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 the "*Composite Adapter Configuration*" appendix in the PingFederate *Administrator's manual*).

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 542, 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.

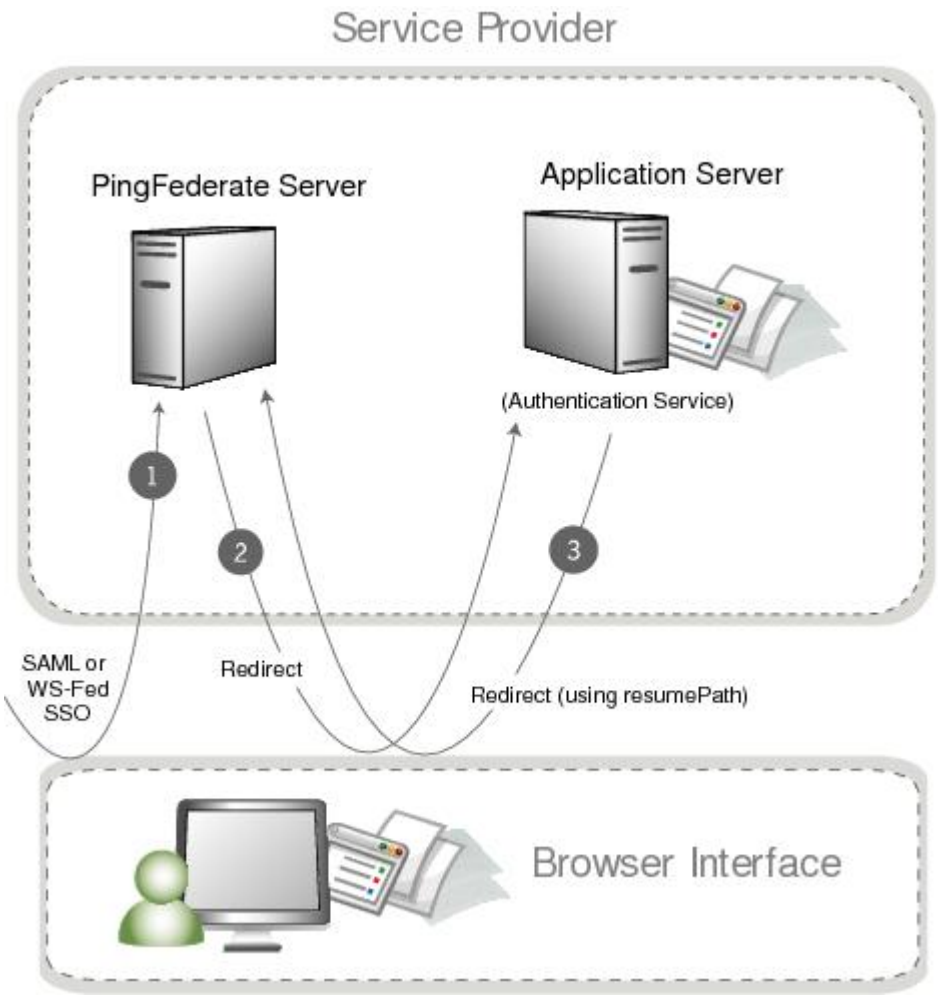
SP account linking

```
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 542, 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 542, 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 542, 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 542, 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 542, 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 password changes in a password credential validator, implement the `com.pingidentity.sdk.password.ChangeablePasswordCredential` 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*).

For each Identity Store Provisioner implementation, in addition to the methods described under *Shared interfaces* on page 542, 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 551.

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 542, 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 541).

   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
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:
  - `pingfederate/server/default/lib`
  - `pingfederate/lib`
  - `pingfederate/sdk/lib`

  - `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 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.
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 **SP Configuration** > **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.

Upgrade considerations

Several specific modifications made since PingFederate 6.11 may affect existing deployments, as described in the following sections.

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 clustering properties* in the PingFederate *Server clustering guide*.

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** > **Digital Signing & XML 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 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.

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* in the *PingFederate Upgrade Utility user guide*.

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* in the *PingFederate Upgrade Utility user guide* for instructions.

If you would like to write logs to a database server, a JDBC 4.1 (or higher) driver is required (see *Write logs to databases* in the PingFederate *Administrator's manual*).

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*, part of the *PingFederate Upgrade Utility user guide*, 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 multi-valued attributes in WS-Federation assertions

Starting in PingFederate 7.3, multi-valued 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 pingfederate_access_grant table, see *Review database changes* in the *PingFederate Upgrade Utility user guide*.

A new database table for OAuth persistent grant extended attributes

Starting in 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`.

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 configuration* in the PingFederate *Administrator's manual*.

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 pingfederate_access_grant table, see *Review database changes* in the *PingFederate Upgrade Utility user guide*.

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.

Multi-valued 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 multi-valued 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.

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. 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)* in the PingFederate *Administrator's manual*).

Deprecated features

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 (WS-Trust STS use cases) 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* in the PingFederate *Upgrade Utility User Guide*.

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 “Provisioning monitoring” under *Runtime monitoring using JMX* in the PingFederate *Administrator's manual* for more information).

LDAP Adapter

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

Known issues and limitations

Administrative console and administrative API

- With respect to managing IdP or SP connections, the administrative API connection support is currently limited to Browser SSO and SLO connections. As a result, it is possible to lose settings that are not supported by the administrative API (for examples, WS-Trust and provisioning) when these settings are added through the administrative console (the UI) and the connection is subsequently modified through the API.
- PingFederate's API documentation is not fully supported in Microsoft Internet Explorer 9 (IE9). The response headers do not show up in the sandbox for API documentation, due to limited support for cross-origin resource sharing (CORS) in IE9.
- The **Save Draft** feature is not available for connections that override plug-in settings from within the connection. In this case, the following DEBUG log message may be generated: `DEBUG [DraftManagerImpl] There was a NotSerializableException trying to serialize the draft`. This message can be safely ignored.

- 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 to the Main Menu 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 Main Menu 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.
- When an nShield Connect HSM in a HA nShield Connect cluster is shutdown, users will receive exceptions in the console when trying to create private keys for both digital signatures and SSL. Before creating new private keys, the nShield Connect HSM should be restored to a normal state (up on the network up, HSM up with OCS cards in their slots), and PingFederate should be restarted.

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 Adapter
  - 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
  - X.509

  Note that this list is updated as other adapters are modified, tested, and released.

RADIUS NAS-IP-Address

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

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.
- For a scenario involving SP-initiated SLO with multiple SPs in which the initiating SP is using a SOAP binding and the other SPs are using one of the front-channel bindings (Artifact, Redirect, or POST) along with a front-channel adapter within the IdP, logout with the front-channel adapter fails. When logout fails with the adapter (a technical limitation, because with SOAP-based SLO, the server does not have access to the browser to kill a session established with a front-channel adapter), any other IdP adapters that are configured for the

connection, and for which logout needs to occur will not be invoked for logout. This includes back-channel (for example, SOAP-based) adapters.
- 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.

WS-Trust STS

- When PingFederate is acting as a WS-Trust STS, if it receives a request on the STS endpoint with the namespace element set to an invalid value (for example, with a trailing slash), it does not normalize this to the valid namespace (for example, without the trailing slash) and fails the transaction. In this case, the workaround is to have the client set the namespace element to the valid namespace.

OAuth

- 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.
- OpenID Connect single-logout is not supported when the IdP session is created through OAuth attribute mapping.

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.
- PingFederate cannot simultaneously log the audit log to multiple databases or HP ArcSight CEF syslog. The audit log can only use a single Log4j 2 appender. For more information, see the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file.
- PingFederate can enforce the masking of sensitive attribute values only within its own code base. External code such as adapter implementations and other product extensions may log attribute values in the clear even when they have been designated to be masked in the GUI. If sensitive attribute values are a concern when using such components, the logging level for the specific component can be adjusted in the `<pf_install>/pingfederate/server/default/conf/log4j2.xml` file to the appropriate threshold to prevent attribute values from appearing in log files.

Configcopy

- If you are using `configcopy` to copy all connections, channels, data sources, adapters, or token translators and you choose to set override properties, the override is applied to all instances. It is recommended that you use care when applying overrides for copy-all operations.
- The `configcopy` tool supports copying only a single reference for each of the following that are defined for a given connection: adapter, data source, Assertion Consumer Service URL, Single Logout Service URL, and Artifact Resolution Service URL. If you have multiple adapters, data stores, or any of the aforementioned service URLs 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 you choose to set an override parameter for a parameter that does not exist in the source configuration, 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 using `configcopy` 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 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. • 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 Settings** > **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 (PKCE) 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 Settings** > **Client Management** screen, pressing the Enter key in the search field no longer returns the administrator to the **OAuth Settings** menu.

Ticket ID	Description
PF-12141	The **OAuth Settings** > **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* 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.

Ticket ID	Description
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** 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.
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.

Ticket ID	Description
	For more information, see *Attribute mapping with multiple data sources*.
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.
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.

Ticket ID	Description
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 workflow (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. Refer to the Security Advisory SECADV011 on the *Customer Portal* for more information
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.
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 wizard fails when PingFederate is deployed behind a web proxy server.

Ticket ID	Issue
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 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
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

Ticket ID	Issue
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 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 (RFC 7009); 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 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 Authorization Server 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 logon 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

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 Authorization 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 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 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.

- Multi-valued 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 Utility user guide

This Upgrade Utility is a command-line tool that automatically migrates existing PingFederate® 6.x (or more recent) configurations to the current version. The tool first installs the new distribution and then copies over files or property values from the previous installation.

Note: The Upgrade Utility does *not* affect the existing installation of PingFederate.

As an option, you can run the tool with a *custom* switch that allows you to choose whether to override any new system defaults or to install newer versions of bundled adapters (see *About custom mode* on page 592).

The Upgrade Utility creates a detailed log showing what files and settings are affected during the upgrade.

If you are upgrading from PingFederate 5.x, contact *support@pingidentity.com* for more information.

Intended Audience

The PingFederate Upgrade Utility and this document are intended for system administrators with significant knowledge of the current PingFederate configuration and enterprise deployment.

Upgrading PingFederate

Fully upgrading your existing PingFederate installation involves initial preparation before running the Upgrade Utility. Then afterward, some manual copying may be needed for customized files or supporting Web-application deployments.

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.

The Upgrade Utility is located in the `/bin` directory of the distribution ZIP as either a batch file (for Windows installations) or a shell script (for UNIX/Linux installations).

Note: The Upgrade Utility does not migrate OAuth clients that are created from PingFederate 6.5 through 6.7 or stored in the internal HyperSQL/Hypersonic database since PingFederate 6.8. Use any of the following interfaces to reconfigure your OAuth clients:

- The **OAuth Settings** > **Client Management** screen in the PingFederate administrative console (see *Manage OAuth clients*).
- The `/oauth/clients` administrative API endpoint (see *PingFederate administrative API*).
- The REST-based Web Service for OAuth client management. This web service requires the client records to be stored in a database. For more information, see *Define an OAuth client data store* and *OAuth Client Management Service*.

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.

Important: Specific modifications made since PingFederate 6.11, such as security updates related to XML encryption, changes in LDAP filter, and updates/upgrades of third-party library files, may affect existing deployments. For detailed information, review *Upgrade considerations* on page 559 in the PingFederate *Release notes*.

Integration Kits

The Upgrade Utility copies Adapters, Connectors, Token Translators, and Integration Kits, from the previous installation to the new installation; the tool does not upgrade them automatically. You may download newer versions by visiting *pingidentity.com* (`www.pingidentity.com/en/products/downloads.html`) and upgrade the kits manually.

Note: Documentation for Integration Kits are available at *documentation.pingidentity.com*.

PingFederate installer

Depending on the source installation, you may also upgrade PingFederate by using the platform-specific installer. The conditions are described as follows:

Source installation	Possible upgrade paths on a Windows server	Possible upgrade paths on a Red Hat Enterprise Linux server	Upgrade path on a UNIX/ Linux server
PingFederate 6.x to 8.x installed by using the PingFederate product ZIP file	PingFederate Upgrade Utility	PingFederate install script PingFederate Upgrade Utility	PingFederate Upgrade Utility
PingFederate 8.x installed by using the PingFederate platform-specific installer	PingFederate installer for Windows PingFederate Upgrade Utility	PingFederate install script PingFederate Upgrade Utility	PingFederate Upgrade Utility

You may download the PingFederate platform-specific installers from *https://www.pingidentity.com/en/products/downloads/pingfederate/platform.html*.

Prepare for the upgrade

Before running the Upgrade Utility, complete the following tasks:

- If you have not already done so, download the current version of the PingFederate ZIP file and obtain a new license key.
- If you have not recently updated the Oracle Java SE Runtime Environment (Server JRE) to version 8 on your PingFederate host machine(s), you must do so.

  Note: When you upgrade the Server JRE, be sure to modify the previously defined paths for the system JAVA_HOME and PATH environment variables. (For more information, see *Install Java* on page 17 in the "Installation" chapter of the *Get started with PingFederate* guide.)

  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.
- Optional: Complete any unfinished connections ("Drafts") in the existing PingFederate administrative console, *if* you want to include them in the migration.
- Unzip the Upgrade Utility distribution into a directory on the same machine running PingFederate.
- If you are upgrading PingFederate in a clustered-server environment, unzip the Upgrade Utility on all the PingFederate servers.

Run the PingFederate Upgrade Utility

Follow these steps to upgrade your PingFederate server using the PingFederate Upgrade Utility.

If you are upgrading multiple PingFederate servers in a cluster, start with the console node.

1. Download the latest version of PingFederate and PingFederate Upgrade Utility at *https://www.pingidentity.com/en/products/downloads/pingfederate/upgrade.html*.
2. Unzip the Upgrade utility ZIP file into a directory on the same machine running PingFederate.
3. Stop PingFederate.
4. On the command line, change the current directory to the `bin` directory of the Upgrade Utility and execute the following command:

Operating system	Command
Windows	`upgrade <pf_install_source> <outputDir> <pingfederateZip> [-c]`
UNIX/Linux	`./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 (see *About custom mode* on page 592)

Examples:

Operating system	Command
Windows	`upgrade c:\sso\pingfederate-6.0.0 c:\sso c:\sso \pingfederate-8.1.4.zip` Note that the sample command creates the new PingFederate installation in the directory `c:\sso\pingfederate-8.1.4` (often referred as `<pf_install>` in PingFederate documentation).
UNIX/Linux	`./upgrade.sh /sso/pingfederate-6.0.0 /sso /sso/ pingfederate-8.1.4.zip` Note that the sample command creates the new PingFederate installation in the directory `/sso/pingfederate-8.1.4` (often referred as `<pf_install>` in PingFederate documentation).

After a moment, the command window 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 `upgrade.log` in the Upgrade Utility base directory.

Tip: Rerun the Upgrade Utility as many times as needed to correct any problems.

5. Start the new PingFederate installation and open the administrative console.
6. Follow the on-screen instructions to upload your license file for the new version.
7. Verify configuration settings in the administrative console.

   Note: Access the connection management screens and ensure that automatic validation is run on each connection.
8. If you are upgrading a clustered PingFederate environment:
   a) Stop each engine node and run the Upgrade Utility to upgrade its PingFederate installation.
   b) Start the *new* installation on each node.
   c) Log on to the PingFederate administrative console.
   d) Go to the **Server Configuration** > **Cluster Management** screen.
   e) On the **Cluster Management** screen, ensure all nodes are shown in the list of IP addresses.
   f) Click **Replicate Configuration**.

   Important: The Upgrade Utility *does not* migrate the administrative user-interface (UI) configuration to clustered servers. For more information about cluster management, see the PingFederate *Server clustering guide*.
9. If PingFederate is running as a service, re-install it (for Windows) or reconfigure it (for UNIX/Linux) to use the new PingFederate installation.

Operating system	Instruction
Windows	1. Remove the existing PingFederate service (see *Remove PingFederate from a Windows server*). 2. Install the new PingFederate service (see *Install the PingFederate service on Windows manually*).
UNIX/Linux	Modify the existing symlink to point to the new PingFederate installation. If the older version of PingFederate was installed using the PingFederate install script for Red Hat Enterprise Linux, update the existing symlink so it points to the new PingFederate installation. By default, the symlinks created by the install script are located in the `/usr/local/` directory and are named `pingfederate-<index>`, where <*index*> is a value assigned by the installation program.

For a clustered PingFederate environment, re-install or reconfigure the PingFederate service on all nodes.

About custom mode

The custom-mode feature (invoked with the `-c` option on the command line) allows administrators 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`
- 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 previous versions is also recommended and will not normally require any follow-on configuration changes (see *OpenToken Adapter configuration*/>).

- 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 Password Credential Validator (PCV), which can be used with the HTML Form or HTTP Basic Adapters (see *HTML Form Adapter configuration* and *HTTP Basic Adapter configuration*).

Review post-upgrade tasks

Review the following post-upgrade tasks:

- *Copy customized files or settings* on page 593
- *Review database changes* on page 594
- *Review log configuration* on page 596
- *Migrate to the integrated Username Token Processor* on page 597

It is also important to perform runtime tests to ensure the new PingFederate installation fulfills your existing use cases.

Copy customized files or settings

If you have modified any user-facing Velocity templates and you wish to preserve the customized user experience, you must copy the HTML files over to the new installation manually for each server node (see *Customizable user-facing screens*). The templates are located in the directory:

`<pf_install>/pingfederate/server/default/conf/template`

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"/>`

Similarly, if you have modified any Web-container configuration settings that need to be carried forward, you must make corresponding changes manually in the new PingFederate deployment. For upgrading PingFederate version 6.9 and higher, you can simply copy over the relevant files from the Jetty configuration directory, `<pf_install>/pingfederate/etc`. For existing versions prior to 6.9, 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 jboss-service.xml 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`.

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. If your PingFederate environment connects to one or more database servers, review the following information and make changes accordingly.

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 the engine nodes.

Important: Support for BoneCP as the JDBC connection pool library has been deprecated and will be removed in a future release. We recommend updating to JDBC 4.1 (or higher) drivers.

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.

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.

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.

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's 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.

For more information about OAuth grant data store, see *Define an OAuth grant data store*.

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` database 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.

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.

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.

For more information about outbound provisioning and the internal data store, see *Outbound provisioning for IdPs*.

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 the older version, PingFederate 8.0 starts using Log4j2 after the upgrade process.

If log4j.xml has changed since it was originally installed, the Upgrade Utility makes a backup of the customized log4j.xml file and creates a new log4j2.xml based on default settings. The backup is located in the <pf_install>/pingfederate/server/default/conf directory. The filename is log4j-old-<SourceVersion>.xml, where <SourceVersion> is the version number of the source PingFederate installation.

Upgrading from PingFederate 8.x

If the Upgrade Utility 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 tool does not currently support automatic merging of customizations made to the existing logging configuration. Instead, the Upgrade Utility copies the modified log4j2.xml to the new installation intact and renames 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 determines that the Log4j configuration file (log4j.xml in the <pf_install>/pingfederate/server/default/conf directory) has changed since it was originally installed, the tool copies 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 Utility.

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 to the integrated Username Token Processor

As of PingFederate 7.2, the Username Token Translator (WS-Trust STS use cases) 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. Create an instance of the integrated Username Token Processor (see *Configure a Username Token Processor instance*).

   Tip: In the **Type** screen, select **Username Token Processor** from the list.

   Note: 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.

2. Map the new token processor instance to the applicable WS-Trust STS SP connection in the **IdP Token Processor Mapping** screen (see *Manage IdP token processor mappings*).

   Note: Repeat this step if you have multiple WS-Trust STS SP connections.

3. Test your WS-Trust STS use cases.
4. 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.
5. Delete all token processor instances of the deprecated Username Token Translator in the **IdP Configuration** > **Token Processors** screen (see *Manage token processors*).
6. Remove the `pf-username-token-translator-<version>.jar` file from the `<pf_install>/pingfederate/server/default/deploy` directory on all PingFederate servers.
7. Restart PingFederate on all PingFederate servers.

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 are functional in the new installation.

Known limitations

- Neither the Upgrade Utility nor the platform-specific installers migrates OAuth clients that are created from PingFederate 6.5 through 6.7 or stored in the internal HyperSQL/Hypersonic database since PingFederate 6.8. Use any of the following interfaces to reconfigure your OAuth clients:
  - The **OAuth Settings** > **Client Management** screen in the PingFederate administrative console (see *Manage OAuth clients*).
  - The `/oauth/clients` administrative API endpoint (see *PingFederate administrative API*).
  - The REST-based Web Service for OAuth client management. This web service requires the client records to be stored in a database. For more information, see *Define an OAuth client data store* and *OAuth Client Management Service*.

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.

- As of version 6.5, PingFederate supports Thales nShield Connect HSM configurations. If your PingFederate installation is configured in a clustered environment with 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. For more information, see *Additional steps for server clusters*.
- The OGNL library used by PingFederate was upgraded with version 6.4. If you use OGNL expressions, we recommend retesting the expressions using the PingFederate administrative console.
- As of PingFederate 6.2, TLS renegotiation was disabled by default for all HTTPS listeners/connectors. This security update affects only existing partner connections using WS-Trust STS and those making use of inbound mutual SSL/TLS authentication for the SOAP or artifact SAML bindings. In such cases, administrators must manually reconfigure `run.properties` to enable the secondary HTTPS port (see *PingFederate properties*). In addition, affected connection partners must update their configurations to use the new endpoint.
- The Upgrade Utility does not migrate any data maintained in the PingFederate internal HyperSQL/Hypersonic database or any external database. If, for example, SaaS Provisioning is enabled in the new PingFederate instance using the internal database, it is re-initialized from the provisioning source.

Download documentation

- *PingFederate 8.1.4 Manuals*

Legal information

PingFederate® Product Documentation

Trademarks

Disclaimer

Index

A

access control *105*
account linking
  configuring *329*
account mapping *66*
accounts, administrator *105*
activation
  for IdP connection *375*
  for SP connection *304*
adapters
  configuring
    IdP *238*
    SP *309*
  integration *64*
  mapping URLs to SP *311*
administrative console
  certificate authentication to *108*
  logging on using LDAP *108*
administrators, adding *106*
affiliations, SP *305*
Apache Commons OGNL page *486*, *489*
archiving data *103*
artifact
  lifetime, setting for IdP connections *349*
  lifetime, setting for SP connections *283*
Artifact Resolution Service *36*, *38*
assertion *44*, *47*
Assertion Consumer Service *37*
Assertion Consumer Service URLs *279*
assertions
  content *258*
  lifetime *257*
  mapping attributes to *265*
Assertions *29*
attribute authority (XASP), configuring *352*
attribute contract
  default for SP connection *277*
  IdP connection *330*
  to SP *259*
attribute query
  configuring for IdP connection *352*
  configuring for SP connection *285*
  mapping names *353*
attributes
  mapping
    about *66*
    for SP connection *273*, *287*, *399*, *413*
  masking in log files *71*, *89*
  overview *67*
attributes:
  sources for express provisioning *354*
attribute sources
  custom *149*
  failover (for IdP) *265*
  for IdP *265*
audit logging *80*
Auto-Connect
  about *75*
  activation (IdP connection) *377*
  activation (SP connection) *307*
  allowed IdP domains *377*
  allowed SP domains *308*
  configuring (as an IdP) *306*
  configuring (as an SP) *376*
  metadata lifetime *141*
  metadata signing *141*

B

back-channel settings
  for IdP connections *370*
  for SP connections *290*
backups, administrative console *103*
bindings
  allowable
    for IdP connection *349*
    for SP connection *283*
  configuring
    for IdP connection *327*
    for SP connection *255*
  SOAP for IdP connection *370*
  SOAP for SP connection *290*
buttons, administrative console *27*

C

certificate authority (CA) *160*
certificates
  authority, verifying *75*
  considerations *80*
  digital signing *164*
  expiration notification *130*
  exporting *164*
  for SOAP authentication *80*
  importing *164*
  management *72*
  revocation of *170*
  selecting XML decryption (SP-to-IdP) *375*
  selecting XML encryption (SP-to-IdP) *374*
  signature verification *164*
  SSL *130*, *161*
  SSL client *163*
  trust models for *74*
  verifying *75*
checklist, for federation *79*
common domains *156*
Common Event Format *99*
configuration
  archive *103*
  exporting *103*
  importing *104*
connection list
  for IdP connection *319*
  for SP connection *246*

connections
importing 466
migrating 114
console buttons 27
CSR, importing/exporting 164
custom data stores
filters (IdP connections) 340
filters (SP connections) 413
selecting fields (IdP connections) 340
selecting fields (SP connections) 273

D

data archiving 103
database
filter
for IdP connection 337
for IdP connection 336
for SP connection 268
database filter
for SP connection 269
database filter:
for SP connection 410
data exchange
automated 82
for IdP connection 82
for SP connection 82
data store
compatibility 13
description 70
for attribute query profile 286
introduction 143
JDBC configuration 144
LDAP configuration 145
selecting
for express provisioning 140, 354
for IdP connection 335
for SaaS provisioning 299
for SP connection 267
troubleshooting 501
data store:selecting
for express provisioning 140, 354
decryption keys
for IdP connections 375
for SP connections 295
defederation 67
deployment diagrams 12
digital signatures
about 72
keys and certificates 164
policy 74
digital signing
for IdP connection 371
for SP connection 291
directory service
code example 469
SOAP example 470
discovery, IdP 155
domains
allowed for IdP Auto-Connect 377
allowed for SP Auto-Connect 308

E

email
addresses 130
notifications 130
server configuration 110
encryption, XML 284, 351
endpoints
protocol, SP 317
endpoints:
protocol, IdP 244
endpointsapplication endpoints
IdP 243
IdP application 243
SP 315
SP application 315
entity ID 137
event trigger, for provisioning 358

F

failover
for data stores (IdP) 265
failsafe attribute source 277
federation
checklist 79
choosing 136
defederation 67
ID 82
roles 136
URLs 137
filter, database
for IdP connection 337
for SP connection 269, 410
filter, LDAP
for IdP connection 339
for SP connection 272, 412
for XASP 316

H

hardware requirements 13
heartbeat, server
checking via HTTP 452
host names, virtual 111

I

identity mapping
about 66
IdP configuration 258
SP configuration 329
IdP
adapters 238
connections
activation 375
introduction 308
IdP Discovery
common domain service 156
configuring 155
local domain service 156

selecting *136*
IdPIdPerror message, SLO
default URL *243*
SLO error message *243*
IdP-initiated SLO
configuring
for IdP connection *328*
for SP connection *256*
IdP-initiated SSO
configuring
for IdP connection *327*
for SP connection *256*
IETF BCP 47 *125*
importing connections *466*
importing metadata
for IdP connection *323*
for SP connection *251*
installation
federation server *17*
license key *88*
Linux service *23*
Server JRE *17*
Windows service *23*
installing
java *17*
integration kits *64*

J

java *17*
Java requirements *13*
JDBC configuration *144*
JRE *17*

K

keys
for XML decryption (SP-to-IdP) *375*
for XML encryption (IdP-to-SP) *295*
keys and certificates
digital signature *164*
SSL *163*

L

LDAP
configuration *145*
directory search
for SP connection *270*, *411*
filter *272*, *339*, *412*
SSL *145*
using SSL *145*
LDAP, using for authentication *108*
license key, installing *88*
licensing
email address to notify *130*
Linux
service *23*
log files *89*
logging
administrator audit *91*
files *89*
modes *93*
transaction *93*
using Common Event Format *99*

M

main menu
for IdP connection *318*
for SP connection *245*
mapping
adapters
URLs to SP *311*
attributes
configuring for SP connection *273*, *287*, *399*, *413*
identity *66*, *258*, *329*
masking attributes *71*
masking attributes in logs *89*
message
signing *72*
validation *73*, *80*
metadata
exporting *101*
importing
for IdP connection *323*
for SP connection *251*
providing for Auto-Connect *76*
signing *103*
signing for Auto-Connect *141*
use *81*
migrating connections *114*
monitoring
JMX *131*
provisioning *131*
server *131*

N

name identifiers
SAML 2.0 *258*
WS-Federation *259*
navigation
buttons *27*
notificationslicensing
expiration notification *130*
runtime *130*

O

OAuth
enabling *181*
Object-Graph Navigation Language (OGNL), editing *488*
operating systems *13*

P

password notification *105*
planning checklist *79*
profiles
configuring
for SP connection *255*

for IdP connection *327*
protocol endpoints
IdP *244*
SP *317*
protocols
troubleshooting *505*
protocolsstandards
choosing *136*
provisioning
enabling *327*
event trigger for *358*
provisioning, as SP
choosing data stores for *354*
provisioning, SaaS
choosing internal data store *140*
choosing source data stores for *299*
pseudonym *44*
pseudonyms
unique values for *240*

R

requirements
data store *13*
hardware *13*
Java *13*
operating systems *13*
Roles *28*
runtime
configuration *112*
reporting *131*

S

SAML
selecting *136*
SAML 1.1
SP connection steps *255*, *325*
SAML 2.0
SP connection steps *254*, *325*
SDK *66*
security
back channel *80*
considerations *72*
Serve JRE installation *17*
server
troubleshooting *501*
server HTTP checking *452*
server ID *80*
server monitoring *131*
server status verification *452*
Service Provider *29*
service URL
IdP, for WS-Fed *347*
SP, for WS-Fed *281*
session integration considerations *80*
signature verification *164*
signing XML files *103*
SLO service URLs *282*
SNMP configuration *131*
SOAP *36*
SOAP responder URL *82*
SP
adapters *309*
connections
activation *304*
introduction *238*
default URL *315*
SP-initiated SLO, configuring
for IdP connection *328*
for SP connection *257*
SP-initiated SSO, configuring
for IdP connection *327*
for SP connection *256*
SSL
about *74*
certificates *161*
keys and certificates *163*
using for LDAP *145*
SSO *468*
SSO directory service *468*
starting and stopping the server
Linux service *23*
summary
for IdP connection *375*
for SP connection *304*
synchronization
clock *80*
provisioning *140*
system administration (single or multiple users) *130*
system requirements *13*

T

time synchronization *80*
transaction logging *80*, *93*
transport security considerations *80*
trigger, for provisioning *358*
troubleshooting
data store *501*
protocols *505*
server *501*
trust models *74*

U

uninstalling *25*
unique IDs *80*
unique values, for pseudonym creation *240*
upgrading PingFederate *103*
user management
setting *130*
users
account management *105*
adding *106*

V

validating messages *80*
verifying certificates *75*
virtual host names *111*
virtual IDs
for IdP servers *246*

for SP servers *319*

W

Web Services
connection management *466*
SSO Directory *468*
Web templates *120*
where clause
for IdP connection *337*
for SP connection *269*, *410*
Windows service, installation *23*
WS-Federation
setting realm ID *137*
SP connection steps *255*, *325*
SP service URL *281*, *347*
WS-Federationroles, choosing IdP/SP
selecting *136*
WSS *46*, *47*
WS-Trust STS
configuring request parameters *386*
enabling *378*
IdP connections *404*
mapping request parameters *389*
SP connections *386*

X

X.509 certificates *72*
XASP:requester mapping
requester mapping *316*
XML encryption
about *75*
IdP-to-SP *284*
SP-to-IdP *351*
XML files, signing *103*