PingAccess

Contents

Release Notes........ 8

PingAccess Release Notes........ 8

Release Notes........ 8

PingAccess Policy Migration Release Notes........ 13

Release Notes........ 13

Ping Identity Automated Deployment for AWS Release Notes........ 16

Release Notes........ 16

About PingAccess........ 16

PingAccess overview........ 16

What is PingAccess?........ 17

PingAccess for Azure AD........ 17

What can I do with PingAccess?........ 18

How does PingAccess work?........ 20

WAM session initiation........ 20

Token mediation........ 21

What can I configure with PingAccess?........ 22

Installing PingAccess........ 25

Install PingAccess........ 25

Installation requirements........ 26

Install PingAccess on Linux........ 30

Install PingAccess on Red Hat Enterprise Linux........ 30

Install PingAccess on Windows........ 31

Start PingAccess........ 32

Access the admin console for the first time........ 33

Access the PingAccess administrative API........ 34

Access the interactive administrative API documentation........ 34

Change configuration database passwords........ 34

Stop PingAccess........ 35

Run PingAccess as a service........ 35

Configure PingAccess to run as a Linux systemv service........ 36

Configure PingAccess to run as a Linux systemd service........ 36

Configure Multiple Instances of PingAccess as Linux services........ 37

Remove the PingAccess Linux service........ 37

Configure PingAccess to run as a Windows service........ 38

Remove the PingAccess Windows service........ 38

Uninstall PingAccess........ 38

Configuring and Customizing PingAccess........ 39

Configure session management........ 39

Configure session management........ 39

Configure server-side session management........ 39

Configure logging........ 41

Configure logging........ 41

Security audit logging........ 41

Logging........ 44
Configure log levels........ 45
Configure a class or package log level........ 45
Enable cookie logging........ 46
Append log messages to syslog and the console........ 46
Write logs to other formats........ 46
Customize and localize PingAccess........ 50
Customization of user-facing pages........ 50
Localization of user-facing pages........ 52

Upgrading PingAccess........ 53
Upgrade PingAccess........ 53
Upgrading your environment........ 53
Upgrade a PingAccess standalone version........ 53
Upgrade a PingAccess cluster........ 55
Upgrade Windows using the installer........ 56
Upgrade RHEL using the installer........ 57
Perform post-upgrade tasks........ 58
Restore a PingAccess configuration backup........ 66
Upgrade utility configuration file reference........ 67
PingAccess: Zero Downtime Upgrade........ 67
Introduction........ 67
Step 1: Disable key rolling........ 68
Step 2: Upgrade the Admin node........ 69
Step 3: Upgrade engines........ 70
Step 4: Enable key rolling........ 74
Recovering from a failed upgrade........ 75

Reference Guides........ 75
API Endpoints........ 75
PingAccess endpoints........ 75
Clustering Reference Guide........ 79
Clustering........ 79
Configure a PingAccess cluster........ 82
Configuring the administrative node........ 83
Configure runtime state clustering........ 84
Configuring the replica administrative node........ 84
Manual fail over to the replica administrative node........ 85
Reinstate a replica administrative node after failing over........ 86
Configure an engine node........ 86
Editing an engine node........ 87
Revoke access from an engine node........ 87
Removing an engine node........ 88
Configuration File Reference Guide........ 88
Configuration file reference........ 88
Deployment Reference Guide........ 100
PingAccess deployment guide........ 100
Use cases and deployment architecture........ 100
Configuration by use case........ 103
Web Access Management........ 106
API Access Management proof of concept deployment architecture........ 112
API Access Management production deployment architecture........ 113
Auditing and proxying proof of concept deployment architecture........ 114
Auditing and proxying production deployment architecture........ 115

Groovy Development Reference Guide........ 117
Groovy........ 117
Groovy Scripts........ 117
Body object........ 119
Exchange object........ 120
Headers object........ 121
Identity object........ 123
JsonNode object........ 124
Logger object........ 125
MediaType object........ 126
Method object........ 126
OAuth Token object........ 127
PolicyContext object........ 127
Request object........ 128
Response object........ 129
SslData object........ 130
Groovy script examples........ 130
Matchers........ 132
Performance Tuning Reference Guide........ 136
Performance tuning........ 136
Java tuning........ 136
Operating system tuning........ 137
Garbage collector configuration........ 140
Acceptor threads........ 140
Worker threads........ 141
Backend server connections........ 141
Logging and Auditing........ 142
Agent tuning........ 142
PingAccess User Interface Reference Guide........ 143
PingAccess User Interface Reference Guide........ 143
Applications........ 143
Sites........ 152
Agents........ 158
Rules........ 161
Access........ 182
Networking........ 194
Security........ 200
System........ 205

Agents and Integrations........ 227
PingAccess for AWS........ 227
PingFederate Environment Requirements for PingAccess for AWS........ 227
Solution Setup Guide for PingAccess for AWS........ 234
Configure an Application for PingAccess for AWS........ 248
PingAccess for Azure AD........ 254
Solution overview........ 254
Configure PingAccess to use Azure AD as the token provider........ 254
Configure a PingAccess application........ 255
Getting started with PingAccess for Azure AD........ 257
PingAccess for Azure AD: Configure dual internal and external secure access........ 258
PingAccess Agent for Apache (RHEL)........ 258
Introduction........ 258
System Requirements........ 259
Installation........ 260
Configuration........ 269

- Troubleshooting.... 273
- Release Notes.... 274

PingAccess Agent for Apache (SLES)....275
- Introduction.... 275
- System Requirements....276
- Installation....276
- Configuration....277
- Troubleshooting.... 282
- Release Notes.... 283

PingAccess Agent for Apache (Windows).... 283
- Introduction.... 283
- System requirements....284
- Installation....285
- Configuration....285
- Release notes....291

PingAccess Agent for IIS....291
- Introduction.... 291
- System Requirements....292
- Installation....293
- Configuration....294
- Troubleshooting.... 299
- Release Notes.... 305

PingAccess Agent for NGINX.... 306
- Introduction.... 306
- System Requirements....307
- Installation....307
- Configuration....308
- Release Notes.... 313

PingAccess Agent Protocol....313
- PAAP Overview.... 313
- PingAccess agent protocol flow.... 314

PingAccess Agent SDK for C.... 328
- Preface....328
- Introduction.... 328
- Getting Started with the PingAccess Agent SDK for C....330
- Release notes....331

PingAccess Agent SDK for Java.... 331
- Preface....331
- Introduction.... 332
- Agent SDK directory structure....333
- Agent SDK prerequisites.... 333
- How to install the servlet filter sample....334
- PingAccess Agent SDK for Java Release History.... 335

PingAccess Addon SDK for Java.... 336
- Preface....336
- Introduction.... 336
- Get started with the SDK.... 337
- Create your own plugins.... 340
- Integrating with third-party services....342
- Implementation guidelines....345
- PingAccess Addon SDK for Java Migration Guide.... 347

Protect a web-based application using PingFederate and PingAccess in a proxy deployment.......377
- Protect a web-based application in a proxy deployment....377
- Configure PingFederate for PingAccess connectivity.... 378
- Connect to PingFederate and configure an application in PingAccess....382
- Test the configuration....385

Protect applications using PingAccess and PingOne for Customers....385
Solution overview....385
Configure PingAccess to use PingOne for Customers as the token provider....386
Configure a PingAccess application....386

Migrating Access Management Infrastructure to PingAccess....388
Introduction....388
About PingAccess Policy Migration (PAPM)....388
Platform architecture....388
Automating policy management....388
PAPM naming process....390
PingAccess Policy Migration....391
Installation and setup....392
Installation....392
Setup....395
Upgrade PingAccess Policy Migration....395
Settings....396
Environment management....396
License management....399
Application settings....399
Database management....400
SSL certificates....402
Version....403
Monitoring Dashboard....403
Summary health check....403
Response times....404
Memory usage....404
Sync status....405
Search....405
Manage monitors....405
Policy Automation....407
Policy promotion process overview....407
Policy Operation....408
Policy Settings....411
Compare Exports....415
Audit Data....416
Policy Testing....416
Architecture overview....417
Settings....417
Logic testing....418
Processing steps....418
View test results....419
Load testing....420
View load test results....420
Managing error messages....424
Policy Migration for CA SSO (Siteminder)....424
Migration overview....424
Migration process....425
Policy Migration for Oracle Access Manager....430
Migration overview....430
Migration process....433
Disaster recovery....439
Set up disaster recovery (data backup)....439
Troubleshooting....440
PAPM logging....440

Login issues....................................440
Policy Automation....................................440
Policy Test Tool....................................441
Policy Migration Tool for CA SSO....................................442
Policy Migration Tool for OAM....................................442

Release Notes

PingAccess Release Notes

Release Notes

PingAccess is a centralized point of security and access control for Web applications and APIs, serving applications and other resources to clients outside an organization while still protecting internal interfaces from unauthorized access. PingAccess protects applications and APIs, enabling access control and identity-based auditing on incoming requests. Featuring a lightweight, highly scalable architecture, PingAccess complements PingFederate with centralized session management and URL-level authorization.

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

PingAccess 5.3.4 - February 2022
These enhancements and issue fixes are included in PingAccess 5.3.4, released in February 2022.

PingAccess 5.3.4 is a cumulative maintenance release for PingAccess 5.3, which introduced several new features, including performance enhancements, logging improvements, simplified PingFederate configuration, and greater control over trusted certificate groups, along with several other enhancements. For more information, see the release notes for *PingAccess 5.3 - August 2019* on page 10.

Resolved issues

Ticket ID	Description
PA-14609	PingAccess upgraded to Log4j version 2.17.1

PingAccess 5.3.3 - September 2021
These enhancements and issue fixes are included in PingAccess 5.3.3, released in September 2021.

PingAccess 5.3.3 is a cumulative maintenance release for PingAccess 5.3, which introduced several new features, including performance enhancements, logging improvements, simplified PingFederate configuration, and greater control over trusted certificate groups, along with several other enhancements. For more information, see the release notes for *PingAccess 5.3 - August 2019* on page 10.

Resolved issues

Ticket ID	Description
PA-14095	Fixed a potential security issue.
PA-14104	Fixed potential `http` request smuggling headers manipulation security issue.

PingAccess 5.3.2 - November 2019
These enhancements and issue fixes are included in PingAccess 5.3.2, released in November 2019.

PingAccess 5.3.2 is a cumulative maintenance release for PingAccess 5.3, which introduced several new features, including performance enhancements, logging improvements, simplified PingFederate configuration, and greater control over trusted certificate groups, along with several other enhancements. For more information, see the release notes for *PingAccess 5.3 - August 2019* on page 10.

Resolved issues

Ticket ID	Description
PA-10898	Fixed an issue causing the Admin API to sometimes use an incorrect proxy configuration.
PA-11938 / PA-11912	Fixed a set of issues with OIDC and token providers after a configuration import or upgrade.
PA-11847	Updated PingAccess to automatically not set the SameSite attribute in additional browsers.
PA-11941	Fixed an issue causing PingAccess to incorrectly process `100 Continue` HTTP responses.
PA-11674	Fixed an issue causing PingAccess authentication to break after a PingFederate upgrade.
PA-11857	Fixed an issue causing slow performance in API application calls.
PA-11821	Specified a value for the `LimitNOFILE` property in the `pingaccess.service` file.

PingAccess 5.3.1 - October 2019

These enhancements and issue fixes are included in PingAccess 5.3.1, released in October 2019.

PingAccess 5.3.1 is a cumulative maintenance release for PingAccess 5.3, which introduced several new features, including performance enhancements, logging improvements, simplified PingFederate configuration, and greater control over trusted certificate groups, along with several other enhancements. For more information, see the release notes for *PingAccess 5.3 - August 2019* on page 10.

Resolved issues

Ticket ID	Description
N/A	Fixed potential security issues.
PA-11503	Updated the handling of the SameSite cookie attribute to avoid browser-specific issues. This change also ensures that Safari 12 will work normally when the SameSite attribute is set to None in a web session. A setting of None will be required when Chrome 80 is released.
PA-11566	Fixed an issue causing upgrades to time out and fail in large environments.
PA-11508	Fixed an issue that prevented PingAccess from starting on a Windows system with a version of OpenJDK 11 without minor releases.
PA-11576	Fixed an issue causing escaped characters to be re-escaped in web session attribute names.
PA-11594	Fixed an issue causing the target system in an upgrade to use default heap space settings, which can slow the upgrade in large environments.
PA-11465	Fixed an issue that caused an upgrade to fail when the same directory is specified as the source and target during an upgrade.
PA-11258	Fixed an issue that caused values of `0` not to display in the user interface.
PA-11528	Fixed an issue causing the `AUDIT.targethost` variable to be set even when the target server did not respond.
PA-11545	Fixed an issue that caused the SameSite cookie attribute to be dropped when a rewrite cookie rule was used.

Ticket ID	Description
PA-11703	Fixed an issue that caused the pingaccess.log to exclude details about Groovy syntax errors.
PA-11656	Fixed an issue that caused the admin console database to temporarily grow significantly larger during startup.
PA-11712	Fixed an issue causing the API audit log to incorrectly report the round trip time for API calls.

PingAccess 5.3 - August 2019

These enhancements and issue fixes are included in PingAccess 5.3, released in August 2019.

Enhancements

Performance improvements

Processing and response speed has been improved for Administrative API calls, engine node and replica administrative node replication, configuration imports, and configuration exports.

Assign a trusted certificate group to a port

You can now assign a trusted certificate group to a specific port. This lets you require client certificate authentication for all connections made to a specific engine listener. See *Defining an engine listener* for more information.

Improved run.sh script compatibility

The `run.sh` script has been enhanced to work with environments such as Alpine Linux and busybox.

Addon SDK enhancement for third-party services

The Addon SDK now lets a plugin declare that it needs a third-party service handle for an endpoint in the OIDC metadata. See *the OIDC Provider Javadoc* for more information.

UI enhancement for resource creation

When you view resources, a list of global unprotected resources is displayed, making it easier to avoid conflicts.

Admin API audit logging improvement

The audit logs for the Admin APIs now include response times.

Startup logging improvement

When PingAccess is started from the command line, the `pingaccess.log` file now includes the startup time.

Upgrade utility logging improvement

The upgrade utility now correctly displays warnings from the APIs.

Configure PingFederate as token provider using metadata

You can configure PingFederate as a token provider using information from the OIDC metadata, meaning that much less information must be entered manually. See *Configuring PingFederate runtime* on page 217 for more information.

View and refresh OIDC metadata

You can view and refresh the metadata for a configured token provider. See *Configuring PingFederate runtime* on page 217, *Configuring PingOne* on page 223, and *Configuring OpenID Connect* on page 223 for more information.

JWT identity mapping exclusion lists

You can now configure a JWT identity mapping to exclude the attributes you specify and include all others. See *Creating a JWT identity mapping* on page 184 for more information.

Added support for Coretto 8 and 11

Coretto 8 and Coretto 11 are now supported Java runtime environments.

View token provider OIDC metadata in PingAccess

This feature allows administrators to view OIDC metadata that is available from the configured token provider.

Import PingAccess license using Admin API

Import a PingAccess license via a **PUT** method on the `/license` endpoint.

Web Session Scope rule

A new rule that examines the contents of the OpenID provider validation response and determines whether to grant access to a back-end target Site based on a match found between the scopes of the validation response and scope specified in the Web Session Scope Rule.

Auditing for Unknown Resources and Redirects

The PingAccess UI and Admin API now support enabling or disabling auditing for redirects and unknown resources. This function was previously managed by the `pa.auditing.unknown.resource` property formerly available in `run.properties`.

Log HTTP request and response headers for clients and applications

Easily modify audit log configuration to include details for HTTP request and response headers for clients and application.

Resolved issues

Ticket ID	Description
N/A	Fixed potential security issues.
N/A	Fixed multiple upgrade utility performance issues.
CC-1004	Updated some clustering terminology for clarity. The optional subclustering feature has been renamed to runtime state clustering.
PA-11140	Fixed an issue causing the replica administrative node to replicate incorrect key data after being promoted to administrative node.
PA-11076	Clarified the error messages displayed when an application configuration does not load correctly.
PA-10240	Reduced the amount of time taken by a PingAccess database backup. Users encountering a slow login process can disable automatic backups by editing the `pa.backup.filesToKeep` parameter as described in the *Configuration file reference guide*.

Ticket ID	Description
PA-11154	Fixed an issue that caused an incorrect Key Pair to be selected for TLS connections to virtual hosts in some cases.
PA-10239	Fixed an issue causing the Admin UI to time out during large configuration imports.
PA-10314	Fixed an issue causing large configuration imports to fail.
PA-10234	Fixed an issue causing large configuration imports to repeatedly restart.
PA-11165	Fixed an issue that caused the heartbeat API endpoint to ignore changes to the reserved application context root.
PA-10925	Fixed an issue that caused the pingfederate API endpoint to ignore hostname properties.
PA-10901	Fixed an issue that prevented old PingFederate metadata from being logged during a metadata update.
PA-10313	Fixed an issue causing excessive database growth on engine nodes.
PA-10829	Removed a Ping copyright statement from the `run.properties` file.
PA-10824	Fixed an issue that made the agent heartbeat endpoint unavailable after a key roll.
PA-10810	Fixed an issue causing invalid OAuth metadata attributes such as client_id to be suggested for attribute rules.
PA-10574	Fixed an issue where the Replica Console Heartbeat endpoint was not responding properly to requests.
PA-10798	Fixed an issue where a web session created in the PingAccess UI may not save correctly if a Client Secret was provided and the login type was specified as `POST`.

Known issues and limitations

Known issues

- Depending on the source version, the upgrade process may change the default settings for the SameSite cookie attribute to make PingAccess cookies work on all browsers. Review the settings for each web session in **Access**# **Web Sessions** to verify that your SameSite cookie attribute values are set to None or Lax, depending on the third-party context needs for PA cookies.
- Use of TLSv1.0 has been maintained for use by legacy versions of Internet Explorer. Since continued use of TLSv1.0 is not recommended for security reasons, users should upgrade to the latest version of Internet Explorer to make use of the more secure TLSv1.1 or TLSv1.2.
- Engines and admin replicas do not connect to admin console if a combination of IP addresses and DNS names are used.
- The token processor can't connect to a JWKS endpoint via SSL when an IP is used rather than a hostname. To workaround this issue, add the hostname as the subject alt name on the key pair.
- When using Internet Explorer 11, you may not be able to view the full application or resource policy because the scroll bar is missing.
- If you create multiple virtual hosts with a shared hostname and associate the hostname with a server key pair, the virtual hosts retain the connection with the server key pair even if they are subsequently renamed. The virtual host must be deleted and recreated to remove the association.

- If PingFederate is configured to use HTTP for runtime endpoints and configured as the token provider, PingAccess does not correctly recognize the endpoint setting and fails to communicate with PingFederate.
- Upgrades will fail if a risk-based authorization rule if a third-party service is not used in the rule.
- Log files may contain excessive warnings issued by Hibernate during startup.
- The Administrative UI can be very slow to load in large environments.
- Configuration importation frequently fails to reconnect to the database on RHEL systems. Affected systems can be restarted by restoring the database from the `/data/archive` directory and retrying the configuration import.
- Asynchronous front-channel logout might fail in some browsers depending on end-user settings. See *https://support.pingidentity.com/s/article/Managing-Single-Log-Out-in-different-browsers* for browser-specific workarounds.

Known limitations

- Internet Explorer and Firefox do not correctly support the HTML5 time tag. When using the Time Range rule, enter time in 24-hour format.
- When installing PingAccess as a Windows service using Windows PowerShell and Java 8, the error message "Could not find or load main class" can be safely ignored.
- Request Preservation is not supported with Safari Private Browsing.
- When using IE 11 to access the PingAccess admin console remotely, a fully qualified domain name or IP address must be specified. For example, `https://console.site.com:9000` and `https://172.17.8.252:9000` will work, while specifying only the host name, `https://console:9000`, will not.
- Incorrect handling for IPv6 literals in Host header. Note that IPv6 is not currently supported.

Upgrade considerations

Several specific changes in PingAccess may require additional steps during an upgrade to the latest version.

Groovy changes

If you created Groovy scripts in PingAccess 4.3 or earlier, your Groovy scripts may use undocumented capabilities that no longer function in PingAccess 5.0 or later.

To check your scripts, you can review your scripts with the current *Groovy Development Reference Guide*, or use a test environment to evaluate the scripts.

To use a test environment:

1. *Install PingAccess 5.0 or later*. This installation is only used for script evaluation purposes.
2. Import your existing groovy scripts to the new environment. You can script this process using the Admin APIs.
3. Review the logs and the JSON error message from the Admin API. If a script is not compatible with PingAccess 5.0 or later, the import fails, and the log or message indicates the cause of the failure.
4. Review the scripts that failed to import into PingAccess 5.0. Scripts that are not compatible with PingAccess 5.0 or later must be evaluated and updated or replaced.

PingAccess Policy Migration Release Notes

Release Notes

PingAccess Policy Migration (PAPM) simplifies Access Management infrastructure migration from CA SSO (Siteminder) and Oracle Access Manager (OAM) to PingAccess. PAPM enables the easy migration of policy and configuration from the existing system to PingAccess, allowing for easier configuration of PingAccess applications and policies while keeping existing policy behavior.

PingAccess Policy Migration 2.0 - December 2018

Release details

- **PingAccess Policy Migration 2.0** is a new major release for PingAccess Policy Migration.
- **PingAccess Policy Migration 2.0** introduces several new features, including:
  - Policy Automation
  - Policy Testing
  - Monitoring Dashboard
- **PingAccess Policy Migration 2.0** removes support for Internet Explorer 11.

Resolved issues

- **PPAT-384**: Fixed an issue where an error was encountered during the import of migration data.

Known limitations

- The uploaded migration file name is hidden when using the IE11 or Edge browsers.
- PingAccess Policy Migration may create rule names that are too long to display in some versions of the PingAccess UI. These rule names may be replaced by an ellipsis.
- An error occurs when adding a PingFederate template if the connection is WS-Trust and browser SSO is not enabled.

Known issues

- PingAccess Policy Migration sets the application context root for migrated applications to `/`. Because each Virtual Host/Context Root combination must be unique, this prevents the creation of more than one application per virtual host.
- Deletion of an import file does not immediately remove the file from the file selection dropdown list.
- If PingAccess or PingFederate servers are not available when migrating configuration, the request may hang for a long time
- Audit record backups are archived as .txt files instead of compressed .csv files.
- PAPM will not migrate an application where the name contains a “/” character.
- UI elements may overlap if the browser width is below 1280 pixels. For optimal user experience, use a browser window size of 1280px x 1024px or larger.

PingAccess Policy Migration 1.0.1 - August 2018

Release details

- **PingAccess Policy Migration 1.0.1** is a cumulative maintenance release for PingAccess Policy Migration 1.0, which was the first release of this product. For information on PingAccess Policy Migration 1.0, see the *release notes*.
- **PingAccess Policy Migration 1.0.1** introduces an *upgrade utility* to allow for easy upgrade of PingAccess Policy Migration to the most recent version.

Resolved issues

- **PPAT-101**: Fixed a potential security issue.

- **PPAT-365**: Fixed an issue where migration would fail if an agent configuration value exceeded 2000 characters.

Known limitations

- The uploaded migration file name is hidden when using the IE11 or Edge browsers.
- PingAccess Policy Migration may create rule names that are too long to display in some versions of the PingAccess UI. These rule names may be replaced by an ellipsis.
- An error occurs when adding a PingFederate template if the connection is WS-Trust and browser SSO is not enabled.

Known issues

- PingAccess Policy Migration sets the application context root for migrated applications to `/`. Because each Virtual Host/Context Root combination must be unique, this prevents the creation of more than one application per virtual host.
- Deletion of an import file does not immediately remove the file from the file selection dropdown list.
- If PingAccess or PingFederate servers are not available when migrating configuration, the request may hang for a long time
- Audit record backups are archived as .txt files instead of compressed .csv files.
- PAPM will not migrate an application where the name contains a “/” character.
- UI elements may overlap if the browser width is below 1280 pixels. For optimal user experience, use a browser window size of 1280px x 1024px or larger.

PingAccess Policy Migration 1.0 - June 2018

Release details

- This is the first release of this product.

Known limitations

- The uploaded migration file name is hidden when using the IE11 or Edge browsers.
- PingAccess Policy Migration may create rule names that are too long to display in some versions of the PingAccess UI. These rule names may be replaced by an ellipsis.
- An error occurs when adding a PingFederate template if the connection is WS-Trust and browser SSO is not enabled.

Known issues

- PingAccess Policy Migration sets the application context root for migrated applications to `/`. Because each Virtual Host/Context Root combination must be unique, this prevents the creation of more than one application per virtual host.
- Deletion of an import file does not immediately remove the file from the file selection dropdown list.
- If PingAccess or PingFederate servers are not available when migrating configuration, the request may hang for a long time
- Audit record backups are archived as .txt files instead of compressed .csv files.
- PAPM will not migrate an application where the name contains a “/” character.
- UI elements may overlap if the browser width is below 1280 pixels. For optimal user experience, use a browser window size of 1280px x 1024px or larger.

Ping Identity Automated Deployment for AWS Release Notes

Release Notes

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

The Ping Identity Automated Deployment for AWS allows you to deploy Ping Identity products to an Amazon Web Services VPC. This solution, both for customers with and without existing AWS resources, allows administrators to easily deploy and configure a functioning clustered environment for your Ping Identity product that includes load balancers, SSH access, and auto scaling functionality. Templates included with the package guide you through the setup of all necessary components, including the S3 bucket, EC2 instances, routing and security parameters, and naming conventions.

Release history

This section shows the release history for the Ping Identity automated deployment for AWS.

Release 1.0.1 - March 2018

- Modified templates to use S3 URLs that include the AWS region.
- Added the **Internal CA AMI ID** as a configurable parameter in the `create-pingaccess.yaml` and `pingaccess-deploy.yaml` templates.
- Added the **PingFederate AMI ID** as a configurable parameter in the `create-pingaccess.yaml` and `pingaccess-deploy.yaml` templates.

Release 1.0.0 - December 2017

- This is the first release of this product. This release supports the deployment of a full PingAccess cluster, as well as an optional PingFederate instance that is available for demonstration purposes.

Known issues and limitations

This section shows the known issues and limitations for the Ping Identity automated deployment for AWS.

Known issues

- There are no known issues in this release.

Known limitations

- The PingAccess for AWS solution cannot yet support agent deployments due to a limitation in the Amazon Elastic Load Balancer implementation.

About PingAccess

PingAccess overview

This document provides an overview of PingAccess. Use this document to gain an understanding of the product, to learn about what you can do, and to discover the many features it provides. To get the most from PingAccess, users should read and understand the concepts included in this document.

As you learn about PingAccess features and functions, review PingAccess scenario documentation for steps to configure them. For a comprehensive set of instructions for using the PingAccess interface, see the *PingAccess User Interface Reference Guide*.

This document answers the following questions:

- *What is PingAccess?* on page 17
- *What can I do with PingAccess?* on page 18
- *How does PingAccess work?* on page 20
- *What can I configure with PingAccess?* on page 22

What is PingAccess?

PingAccess is an identity-enabled access management product that protects Web Applications and APIs by applying security policies to client requests. In simpler terms, PingAccess allows you to protect sites, APIs, and other resources using rules and other authentication criteria. It works in conjunction with PingFederate or other common token provider via the OAuth 2.0 and OpenID Connect protocols to integrate identity-based access management policies via a federated corporate identity store using open standards access protocols.

PingAccess for Azure AD

PingAccess for Azure AD is a free version of PingAccess for users of Microsoft's Azure AD that allows you to protect up to 20 applications. The goal of this solution is to allow for greater control over the access to legacy on-premise applications through the use of PingAccess Identity Mapping functionality.

This free version includes a limited feature set that is intended to support the basic requirements for application protection using this solution. Users of PingAccess for Azure AD are able to upgrade to a full license that will allow the use of the full PingAccess feature set.

Important: When your PingAccess for Azure AD license expires, access to the admin API is removed and you are unable to configure the product. Though managed access to configured applications continues, you must upload a new license file before you can make any additional configuration changes.

Upgrade notice: PingAccess for Azure AD provides a limited feature set that may not be compatible with existing PingAccess configurations. For this reason, upgrading from an earlier full version of PingAccess to PingAccess for Azure AD is **not supported**.

The following table details the available functionality on each of the PingAccess versions, both in the PingAccess user interface and the API.

Functionality	PingAccess	PingAccess for Azure AD
Create applications	Yes	Limited to 20 web session applications.
Create site authenticators	Yes	Limited to Basic and Mutual TLS.
Configure identity mappings	Yes	Limited to Header and JWT.
Create load balancing strategies	Yes	Limited to Header-Based and Round Robin.
Configure web sessions	Yes	Limited to web sessions with OIDC login type CODE.
Configure token provider	Yes	Limited to Microsoft Azure AD authentication source.
Export/Import configuration	Yes	Limited to configurations that includes only features permitted by license type.

Configure policies	Yes	No
Specify authentication requirements	Yes	No
Create and configure custom plugins using the SDK	Yes	No
Configure sites	Yes	Yes
Configure agents	Yes	Yes
Create virtual hosts	Yes	Yes
Configure unknown resource handling	Yes	Yes
Configure availability profiles	Yes	Yes
Configure HTTP request handling	Yes	Yes
Configure listeners	Yes	Yes
Configure forward proxy settings	Yes	Yes
Manage certificates	Yes	Yes
Manage key pairs	Yes	Yes
Configure administrator authentication	Yes	Yes
Configure clustering	Yes	Yes
Manage licenses	Yes	Yes

What can I do with PingAccess?

PingAccess provides a highly customizable solution to identity access management that allows you to control access in a variety of ways by specifying a wide range of conditions that must be satisfied. Read the following sections to discover the methods PingAccess uses to control access and perform system functions. To learn more about the configuration required for any of the following topics, see PingAccess configuration scenarios on *support.pingidentity.com/s/documentation*.

The main functionality of PingAccess is to allow you to protect an application or API. You can:

- Use PingAccess to protect the application and API resources to which client requests are forwarded.
- Partition applications for tighter access control through the use of resources.
- Customize configuration of site authenticators and authentication requirements to suit the security needs of your organization.
- Incorporate legacy authentication mechanisms through Token Mediation.
- Apply policies to define how and when a client can access target resources.

Customize your identity access management configuration with the following features.

Apply policies

Use policies, made up of rules, set of rules, or groups of rule sets applied to an application and its resources, to define how and when a client can access target sites. Rules are the building blocks for access control and request processing.

Backup and restore

Backup or restore a PingAccess configuration with just a few clicks.

Configure a token provider

PingAccess can be configured to use PingFederate as the token provider or may be configured to use a common token provider via the OAuth 2.0 or OpenID Connect protocols.

Configure administrator authentication

Allow administrators to authenticate with a simple username and password, or configure them to authenticate via SSO or API in conjunction with PingFederate.

Configure advanced network settings

Create an availability profile to determine how you want to classify a target server as having failed, configure listener ports, define a load balancing strategy, or use HTTP Requests to match a served resource with the originating client.

Configure logging

Capture several log types, including those for the engine, security auditing, and cookies. Store logs in Splunk, in an Oracle, PostgreSQL, or SQL Server database, or in a file.

Configure Single Logout

End PingAccess sessions easily when used in conjunction with PingFederate managed sessions or compatible third party OpenID Connect providers.

Create clusters

Deploy PingAccess in a clustered environment to provide higher scalability and availability for critical services. Use subclusters to provide better scaling of large PingAccess deployments by allowing multiple engine nodes in the configuration to share information. Place a load balancer in front of each subcluster in order to distribute connections to the nodes in the subcluster.

Customize PingAccess look and feel

Customize and localize the PingAccess pages your users will see, including those for error messages and logout confirmation.

Customize with SDKs

Customize development with SDKs to extend the functionality of the PingAccess server.

Manage certificates and key pairs

Import certificates to establish trust with certificates presented during secure HTTPS sessions. Import or generate key pairs that include the private key and X.509 certificate required for HTTPS communication.

Manage sessions

Use web sessions to define the policies for web application session creation, lifetime, timeout, and scope. Use multiple web sessions to scope the session to meet the needs of a target set of applications. Web sessions improve the security model of the session by preventing unrelated applications from impersonating the end user.

Manually configure runtime parameters

Use a text editor to modify configuration file settings used by PingAccess at runtime.

Protect an application or API

Use PingAccess to protect the application and API resources to which client requests are forwarded. Partition applications for tighter access control through the use of resources. Customize configuration of site authenticators and authentication requirements to suit the security needs of your organization.

Tune performance

Optimize a wide variety of PingAccess components for maximum performance.

Upgrade an existing installation

Easily upgrade an existing installation using the installer, or more carefully manage the upgrade process with the PingAccess Upgrade Utility.

Use APIs

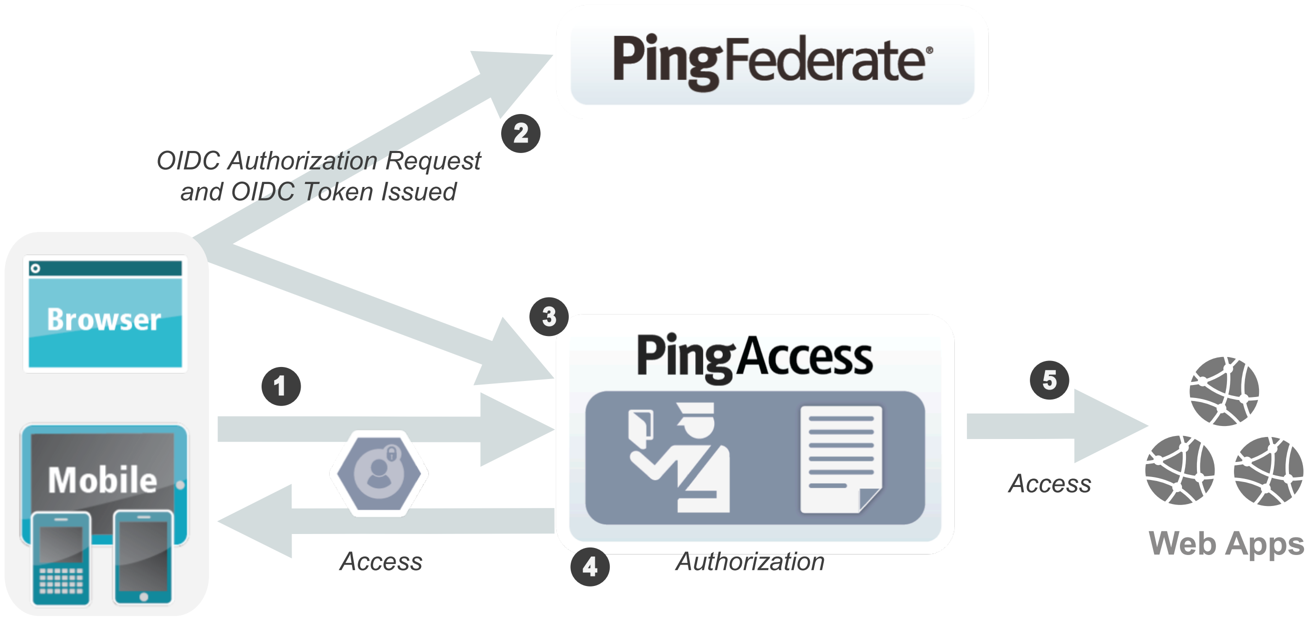
Use the PingAccess APIs to provide a powerful configuration and management experience outside the PingAccess user interface.

How does PingAccess work?

Access requests are either routed through a PingAccess Gateway to the target Site, or they are intercepted at the target web application server by a PingAccess Agent, which in turn coordinates access policy decisions with a PingAccess Policy Server. In either instance, policies applied to access requests for the target Application are evaluated, and PingAccess makes a policy-based decision to grant or deny access to the requested resource. When access is granted, client requests and server responses can be modified to provide additional identity information required by the target Application.

WAM session initiation

When a user authenticates, PingAccess applies the application and resource-level policies to the request. Once policy evaluation is passed, any required token mediation between the back-end Site and the authenticated user is performed. The user is then granted access to the Site.



Processing steps:

1. When a user requests access to a Web resource from PingAccess, PingAccess inspects the request for a PingAccess Token.
2. If the PA Token is missing, PingAccess redirects the user to an OpenID Connect Provider (OP) for authentication.

ⓘ **Info:** When using an OP, an OAuth Client must already be configured in PingAccess. For steps on configuring an OAuth Client within PingFederate, see the *PingFederate Administrator's Manual*. To configure the OAuth Client within PingAccess, see the PingAccess scenario to *Configure a Token Provider*.

3. The OP follows the appropriate authentication process, evaluates domain-level policies, and issues an OpenID Connect (OIDC) ID Token to PingAccess.
4. PingAccess validates the ID Token and issues a PA Token and sends it to the browser in a cookie during a redirect to the original target resource. Upon gaining access to the resource, PingAccess evaluates application and resource-level policies and optionally audits the request.

   ⓘ **Info:** PingAccess can perform *Token Mediation* by exchanging the PA Token for the appropriate security token from the PingFederate STS or from a cache (if token mediation occurred recently).
5. PingAccess forwards the request to the target site.
6. PingAccess processes the response from the site to the browser (step not shown).

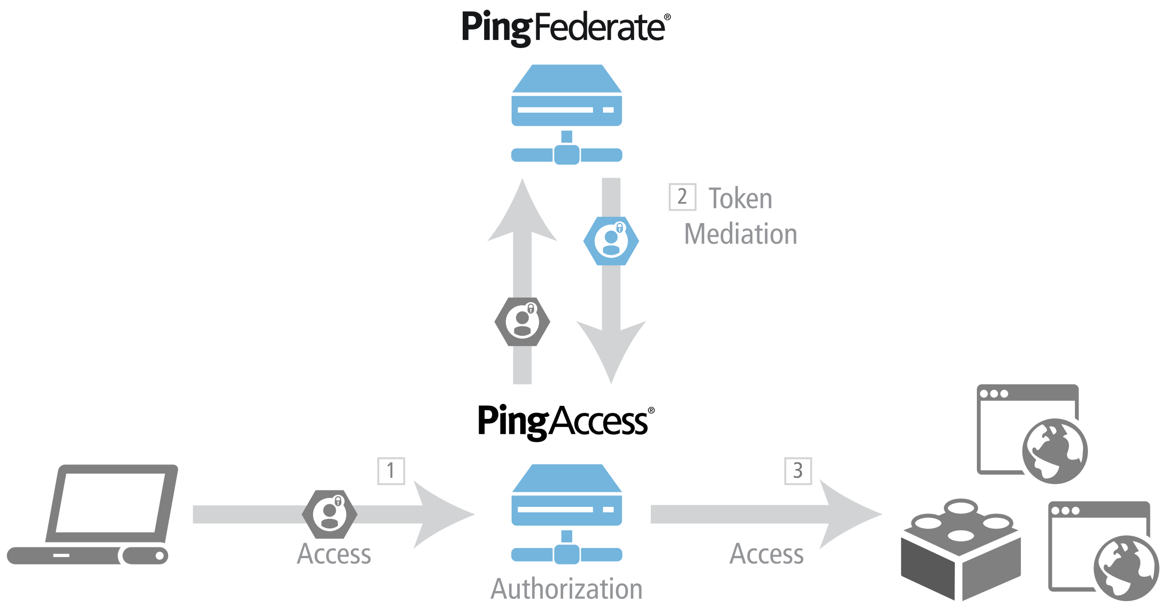
ⓘ **Info:** See the Session Management scenario for more information.

Token mediation

When planning a PingAccess deployment, it is necessary to take inventory of existing applications, and their authentication requirements and mechanisms. When an existing token-based authentication mechanism is in use, retrofitting that mechanism may not always be desirable or cost-effective.

Token Mediation allows a PingAccess gateway to use a PingFederate token generator to exchange the PA Token or an OAuth Bearer Token for a security token used by the foreign authentication system. The access request is transparent to the user, allowing PingAccess to transparently manage access to systems using those foreign tokens. The request is also transparent to the protected application, which handles the access request as if it came from the user directly. Once token mediation has occurred, the token used for accessing the application is cached for continued use during the session.

The following illustration shows an example of token mediation using PingFederate to exchange a PA Token or OAuth Bearer Token for a different security token.



Processing steps:

1. A user requests a Resource from PingAccess with a PA Token or OAuth Bearer Token.

   ⓘ **Info:** This example assumes the user has already obtained a PA Token or OAuth Bearer Token. See the *Session Management* scenario for information on how users authenticate with PingFederate and obtain a PA Token or OAuth Bearer Token.

2. PingAccess evaluates resource-level policies and performs token mediation by acquiring the appropriate security token from the PingFederate STS specified by the Site Authenticator.
3. PingAccess sends the request to the Site (Web application) with the appropriate token.
4. PingAccess returns the response to the client (not shown).

What can I configure with PingAccess?

PingAccess includes a wide range of features that allow you to customize your identity access management deployment. To learn more about these features, read the following descriptions.

Agents

Agents are web server plugins that are installed on the web server hosting the target application. Agents intercept client requests to protected applications and allow or deny the request to proceed by consulting the Policy Manager or using cached information. Agents communicate with the PingAccess Policy Server via the PingAccess Agent Protocol (PAAP) which defines in detail the possible interactions between agents and Policy Server. Agents have a name to identify them and a shared secret to authenticate with to Policy Server. Agents do not need to be unique. There can be any number of agents using the same name and secret and they are all treated equally by Policy Server. This is useful in complex deployments where unique agents would be difficult to manage. Agents can be assigned as the destination for one or more applications by name.

Applications

Applications represent the protected web applications and APIs to which client requests are sent. Applications are composed of one or more resources and have a common Virtual Host and Context Root and correspond to a single target site. Applications also use a common Web Session and Identity Mapping. Access control and request processing rules can be applied to applications and their resources on the Policy Manager page to protect them. Applications can be protected by PingAccess Gateway or PingAccess Agent. In a gateway deployment, the target application is specified as a Site. In an agent deployment, the application destination is an Agent.

Authentication requirements

Authentication Requirements are policies that dictate how a user must authenticate before access is granted to a protected Web Application. Authentication methods are string values and ordered in a list by preference. At runtime, the type of authentication attempted is determined by the order of the authentication methods.

For example, a user attempts to access a PingAccess Web Application configured with an authentication requirement list containing the values (password, cert). PingAccess redirects the user to PingFederate requesting either password or certificate user authentication. PingFederate authenticates the user based on the password and issues an OIDC ID Token to PingAccess (containing the authentication method that was used). PingAccess ensures that the authentication method matched the requirements and redirects the user to the originally requested Application with the PA cookie set. The user navigates to the Application and access is granted. When the user attempts to access a more sensitive Application, configured with an authentication requirement list containing the value (cert), they are redirected to PingFederate to authenticate with a certificate.

If you configure Applications with authentication requirement lists that have no overlap. For example, one list has (password), another list (cert), a user navigating between Applications may be required to authenticate each time they visit an Application. When configuring authentication requirement lists to protect higher value Applications with step-up authentication, consider including stronger forms of authentication when configuring lower value Applications.

Auth token management

Auth token management settings define the issuer and signing configuration used by JWT identity mappings.

Availability profiles

Availability Profiles are used in a Site configuration to define how PingAccess classifies a backend target server as failed. Sites require the selection of an availability profile, even if only one target is provided.

If multiple targets are specified in a site configuration but a load balancing strategy is not applied, then the Availability Profile will cause the first listed target in the site configuration to be used unless it fails. Secondary targets will only be used if the first target is not available.

Certificates

Certificates are used to establish anchors used to define trust to certificates presented during secure HTTPS connections. Outbound secure HTTPS connections such as communication with PingFederate for OAuth access token validation, identity mediation, and communication with a target Site require a certificate trusted by PingAccess. If one does not exist, communication is not allowed.

Certificates used by PingAccess may be issued by a CA or self-signed. CA-issued certificates are recommended to simplify trust establishment and minimize routine certificate management operations. Implementations of an X.509-based PKI (PKIX) typically have a set of root CAs that are trusted, and the root certificates are used to establish chains of trust to certificates presented by a client or a server during communication.

The following formats for X.509 certificates are supported:

- Base64 encoded DER (PEM)
- Binary encoded DER

Clustering

PingAccess can be configured in a clustered environment to provide higher scalability and availability for critical services.

PingAccess clusters are made up of three types of nodes:

Administrative Node

Provides the administrator with a configuration interface.

Replica Administrative Node

Provides the administrator with the ability to recover a failed administrative node using a manual failover procedure.

Engine Node

Handles incoming client requests and evaluates policy decisions based on the configuration replicated from the administrative node.

Any number of clustered engines can be configured in a cluster, but only one administrative console and one replica administrative console can be configured in a cluster.

Further use of subclusters provides better scaling of very large PingAccess deployments by allowing multiple engine nodes in the configuration to share certain information.

HTTP requests

HTTP Requests are used to match a served resource with the originating client when one or more reverse proxies are between the client and the served resource. For example, when a reverse proxy sits between the client and the PingAccess server or a

PingAccess agent, the additional proxy might be identified as the client. Such proxies can be configured to inject additional headers to relay the originating client address.

Identity Mappings

Identity mappings make user attributes available to back-end sites that use them for authentication. There are multiple types of identity mappings, each with different behavior and a distinct set of fields to specify the identity mapping behavior.

Key pairs

Key pairs are required for secure HTTPS communication. A Key Pair includes a private key and an X.509 certificate. The certificate includes a public key and the metadata about the owner of the private key.

PingAccess listens for client requests on the administrative console port and on the PingAccess engine port. To enable these ports for HTTPS, the first time you start up PingAccess, it generates and assigns a Key Pair for each port. These generated Key Pairs are assigned on the HTTPS Listeners page.

Additionally, Key Pairs are used by the **Mutual TLS Site Authenticator** to authenticate PingAccess to a target Site. When initiating communication, PingAccess presents the client certificate from a Key Pair to the Site during the mutual TLS transaction. The Site must be able to trust this certificate in order for authentication to succeed.

Listeners

Listeners monitor ports for incoming requests. PingAccess can place listeners on ADMIN, ENGINE, and AGENT ports.

Load balancing strategies

Load Balancing Strategies are used in a Site configuration to distribute the load between multiple backend target servers. Load balancing settings are optional, and only available if more than one target is listed for a site. This functionality can replace a load balancer appliance between the PingAccess engine nodes and the target servers, allowing for a simpler network architecture.

The Header-Based strategy requires a header be included in the request that defines the target to select from the Site configuration. This strategy has an option to fall back if the requested target is unavailable, or if the header is missing from the request.

The Round Robin strategy has a sticky session option that permits a browser session to be pinned to a persistent backend target. This strategy works in conjunction with the availability profile to select a target based on its availability, and the load balancer will not select a target that is in a failed state.

Policies

The Policy Manager is a rich drag-and-drop interface where you can manage policies by creating Rules, building Rule Sets and Rule Set Groups, and applying them to Applications and Resources. Policies are rules, set of rules, or groups of rule sets applied to an application and its resources. Policies define how and when a client can access target Sites. When a client attempts to access an application resource identified in one of the policy's Rules, Rule Sets, or Rule Set Groups, PingAccess uses the information contained in the policy to decide whether the client can access the application resource and whether any additional actions need to take place prior to granting access. Rules can restrict access in a number of ways such as testing user attributes, time of day, request IP addresses, or OAuth access token scopes. Rules can also perform request processing such as modifying headers or rewriting URLs.

Proxies

Configure settings to authenticate with a forward proxy server when PingAccess makes requests to sites or token providers.

Rules, Rule sets, and Rule set groups

Rules are the building blocks for access control and request processing. There are many types of rules, each with different behavior and a distinct set of fields to specify the rule behavior. Rule Sets allow you to group multiple Rules into re-usable sets which can be applied to applications and resources. Rule set groups can contain rule sets or other rule set groups, allowing the creation of hierarchies of rules to any level of depth. Rule sets and rule set groups can be applied to applications and resources as required.

Sites

Sites are the target applications or APIs which PingAccess Gateway is protecting and to which authorized client requests are ultimately forwarded to.

Site Authenticators

When a client attempts to access a target Web Site, that Site may limit access to only authenticated clients. PingAccess integrates with those security models using Site Authenticators. PingAccess supports a variety of Site Authenticators that range from basic username/password authentication to certificate and token-based authentication. Create a Site Authenticator for the type of authentication the Site requires.

Token provider

Token providers are used as a method of providing credentials for secure access to a given target.

Unknown resources

Unknown resources are resources for which there is no PingAccess definition. You can specify the default and per-agent handling behavior for unknown resource requests and configure custom error responses.

Virtual Hosts

Virtual Hosts enable PingAccess to protect multiple application domains and hosts. A Virtual Host is defined by the host name and host port.

Web sessions

Web Sessions define the policy for Web application session creation, lifetime, timeouts, and their scope. Multiple Web Sessions may be configured to scope the session to meet the needs of a target set of applications. This improves the security model of the session by preventing unrelated applications from impersonating the end user.

Installing PingAccess

Install PingAccess

This document provides instructions to install PingAccess.

PingAccess can be installed on:

- *Linux*
- *Red Hat Enterprise Linux*
- *Windows*

After you install PingAccess, you:

- *Start PingAccess*
- *Access the admin console for the first time*
- *Change configuration database passwords* on page 34

You can also:

- *Stop PingAccess* on page 35
- *Run PingAccess as a service* on page 35
- *Uninstall PingAccess* on page 38

Installation requirements

These sections detail system, hardware, and port requirements for installing PingAccess.

- *System requirements*
- *Hardware requirements*
- *Port requirements* on page 27

System requirements

PingAccess is certified as compatible for deployment and configuration with systems meeting these 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.

ⓘ **Note:** PingAccess supports IPv4 addressing. There is currently no support for IPv6 addressing.

Operating systems

ⓘ **Note:** PingAccess 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 PingAccess server may be affected.

- Microsoft Windows Server 2012 R2 Datacenter
- Microsoft Windows Server 2016
- Red Hat Enterprise Linux ES 6.9
- Red Hat Enterprise Linux ES 7.5
- SUSE Linux Enterprise 11 SP4
- SUSE Linux Enterprise Server 12.3

Docker support

- Version: Docker 18.09.0 CE
- Host operating system: Canonical Ubuntu 16.04 LTS

Virtual systems

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

ⓘ **Info:** 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

- Amazon Coretto 8 (64-bit)
- Amazon Coretto 11 (64-bit)
- Oracle Java SE Development Kit (JDK) 11 (64-bit)

- Oracle Java SE Runtime Environment (Server JRE) 8 (64-bit)
- OpenJDK 11 (64-bit)

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

PingFederate

- PingFederate 8.4
- PingFederate 9.3

Browsers for end users

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

Browsers for admin console

- Google Chrome
- Mozilla Firefox
- Internet Explorer 11 and higher

Audit event storage (external database)

- MS SQL Server 2012
- MS SQL Server 2016
- Oracle 11g R2
- Oracle 12c
- PostgresSQL 9.6.1

Hardware requirements
PingAccess is supported on hardware that meets these requirements.

ⓘ **Info:** Although it is possible to run PingAccess 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.

Although the requirements for different environments will vary, we recommend running PingAccess on hardware that meets or exceeds these specifications:

- Multi-CPU/Cores (8 or more)
- 4 GB of RAM
- 2.1 GB of available hard drive space

Port requirements
This table summarizes the ports and protocols that PingAccess uses to communicate with external components. This information provides guidance for firewall administrators to ensure the correct ports are available across network segments.

ⓘ **Info:** *Direction* refers to the direction of requests relative to PingAccess. *Inbound* requests are requests received by PingAccess from external components. *Outbound* requests are requests sent by PingAccess to external components.

Service	Port details	Source	Description
PingAccess Administrative Console	Protocol: HTTPS Transport: TCP Default port: 9000 Destination: PingAccess Administrative Engine Direction: Inbound	PingAccess Administrator browser, PingAccess administrative API REST calls, PingAccess Replica Admin and clustered Engine nodes	Used for incoming requests to the PingAccess administrative console. Configurable using the `admin.port` property in the `run.properties` file. See the *Configuration file reference guide* for more information. This port is also used by clustered engine nodes and the replica admin node to pull configuration data using the admin REST API.
PingAccess Cluster Communications Port	Protocol: HTTPS Transport: TCP Default port: 9090 Destination: PingAccess Administrative Engine Direction: Inbound	PingAccess Administrator browser, PingAccess administrative API REST calls, PingAccess Replica Admin and clustered Engine nodes	Used for incoming requests where the clustered engines request their configuration data. Configurable using the `clusterconfig.port` property in the `run.properties` file. See the *Configuration file reference guide* for more information. This port is also used by clustered engine nodes and the replica admin node to pull configuration data using the admin REST API.
PingAccess Engine	Protocol: HTTP/HTTPS Transport: TCP Default port: 3000* Destination: PingAccess Engine Direction: Inbound	Client Browser, Mobile Devices, PingFederate Engine	Used for incoming requests to the PingAccess runtime engine. Configurable using the `Listeners` configuration page. See the *PingAccess user interface reference guide* for more information.

Service	Port details	Source	Description
PingAccess Agent	Protocol: HTTP/HTTPS Transport: TCP Default port: 3030 Destination: PingAccess Engine Direction: Inbound	PingAccess Agent	Used for incoming Agent requests to the PingAccess runtime engine. Configurable using the `agent.http.port` property of the `run.properties` file. See the *Configuration file reference guide* for more information.
PingFederate Traffic	Protocol: HTTPS Transport: TCP Default port: 9031 Destination:PingFederate Direction: Outbound	PingAccess Engine	Used to validate OAuth Access Tokens, ID Tokens, make STS calls for Identity Mediation, and return authorized information about a user. Configurable using the `PingFederate Settings` page within PingAccess. See the *PingAccess user interface reference guide* for more information.
PingAccess Cluster Traffic	Protocol: JGroups Transport: TCP Default port: 7610 Destination: PingAccess Engine Direction:Inbound	PingAccess Engine	Used for communications between engine nodes in a cluster. Configurable using the `run.properties` file. See the *Configuration file reference guide* for more information.
PingAccess Cluster Traffic	Protocol: JGroups Transport: TCP Default port: 7710 Destination: PingAccess Engine Direction:Inbound	PingAccess Engine	Used by other nodes in the cluster as part of the cluster's failure-detection mechanism. Configurable using the `run.properties` file. See the *Configuration file reference guide* for more information.
PingAccess Cluster Traffic	Protocol: JGroups Transport: UDP Default port: 7500 Destination: PingAccess Engine Direction: Inbound	PingAccess Engine	Used by other nodes in the same cluster to share information. Configurable using the `run.properties` file. See the *Configuration file reference guide* for more information.

*In addition to port 3000, additional engine listener ports defined in the configuration need to be open as well.

Install PingAccess on Linux

You can install PingAccess on a SUSE Linux system.

Before you begin

Prior to starting the installation, the following prerequisites must be satisfied:

- Ensure the *installation requirements* are met.
- Ensure you are logged on to your system with appropriate privileges to install and run an application.

> **Note:** On Linux, we recommend that you install and run PingAccess as a non-root user.

- A *supported Java runtime* must be installed.
- The System or User Environment Variable JAVA_HOME must exist and be set to a value that represents the location of your Java installation (ie; `usr/java/jdk 1.8.0_74`).
- The JRE `/bin` directory (ie; `usr/lib64/jvm/jre/bin`) path must be added to the `PATH` variable so it is available for scripts that depend on it.
- You must have a `pingaccess.lic` license file.

> **Note:** If you do not have one, you can request an evaluation key at the *Request a License Key page* (`www.pingidentity.com/content/pic/en/products/request-license-key.html`). During the first run of PingAccess, you will be prompted to upload the license file via the UI.

Steps

1. Download the distribution ZIP file.
2. Extract the distribution ZIP file into your installation directory.

Results

> **Tip:** If you are deploying PingAccess in a cluster configuration, see PingAccess cluster configuration documentation.

Install PingAccess on Red Hat Enterprise Linux

You can install PingAccess on a Red Hat Enterprise Linux system.

Before you begin

Prior to starting the installation, the following prerequisites must be satisfied:

- Ensure the *installation requirements* are met.
- Ensure you are logged on to your system with appropriate privileges to install and run an application.

> **Note:** On RHEL, we recommend that you install and run PingAccess as a non-root user.

- A *supported Java runtime* must be installed.
- The System or User Environment Variable JAVA_HOME must exist and be set to a value that represents the location of your Java installation (ie; *usr/java/jdk 1.8.0_74*).
- The JRE `/bin` directory (ie; `usr/lib64/jvm/jre/bin`) path must be added to the `PATH` variable so it is available for scripts that depend on it.

- You must have a `pingaccess.lic` license file.

> ⓘ **Note:** If you do not have one, you can request an evaluation key at the *Request a License Key page* (`www.pingidentity.com/content/pic/en/products/request-license-key.html`). During the first run of PingAccess, you will be prompted to upload the license file via the UI.

Steps

1. Download the installation script.
2. Optional: Download the distribution tar file into your installation directory.

   > ⓘ **Note:** Downloading the distribution file before running the script may save time. If you do not download the distribution file, the installer downloads the required files during the installation process.
3. Launch the installation script: `sudo -E ./pa-install-<version>.sh`

   > ⓘ **Note:** On supported RHEL versions, the newer **systemd** scripting service is used during installation. On earlier versions, the installation is performed using the older **systemv** scripting service.
4. Follow the prompts on the screen, enter your preferred options, or accept the default values.
   The installation completes by starting up PingAccess according to the parameter values you supplied.
   Wait until you see the "Starting" and "Started" messages.
5. Open a browser and enter the machine and PingAccess admin port in the URL, for example:

   `https://`*`yourhost`*`:9000`
6. Finalize your setup.

Results

> ⓘ **Tip:** If you are deploying PingAccess in a cluster configuration, see PingAccess cluster configuration documentation.

Install PingAccess on Windows

You can install PingAccess on a Windows system.

Before you begin

Prior to starting the installation, the following prerequisites must be satisfied:

- Ensure the *installation requirements* are met.
- Ensure that you are logged on to your system with appropriate privileges to install and run an application.

  > ⓘ **Note:** The Windows installer will ask for admin privileges during installation.
- A *supported Java runtime* must be installed.
- The System Environment Variable JAVA_HOME must exist and be set to a value that represents the location of your Java installation (ie; `C:\Program Files\Java\jre 1.8.0_91`).
- The *javapath* directory path (ie; `C:\Program Files\Oracle\Java\javapath`) must be added to the `PATH` variable.

Steps

1. Download the PingAccess Windows installer.
2. Double-click on the icon to launch the PingAccess Setup Wizard.

3. Click **Next** and follow the prompts to complete the installation using the following information for the **Operational Mode** that you select.

Operational Mode	Requirements
Standalone	**Ports** ▪ **PingAccess administrative console:** TCP 9000 ▪ **PingAccess agent protocol:** TCP 3030
Clustered Admin Node	**Ports** ▪ **PingAccess administrative console:** TCP 9000 ▪ **Configuration query port:** TCP 9090
Clustered Replica Admin Node	**Ports** ▪ **PingAccess administrative console:** TCP 9000 ▪ **Configuration query port:** TCP 9090 **Prerequisites** ▪ A Clustered Admin Node must be installed and configured. ▪ A configuration data archive file for the Replica Admin Node must be available. Consult PingAccess clustering documentation for more information. **Note:** We recommend that you install the Clustered Replica Admin Node on a separate machine in the same network.
Clustered Engine Node	**Ports** ▪ **PingAccess agent protocol:** TCP 3030 **Prerequisites** ▪ A Clustered Admin Node must be installed and configured. ▪ A configuration data archive file for the Clustered Engine Node must be available. Consult PingAccess clustering documentation for more information.

4. Copy the URL of the PingAccess admin console that is displayed on the final screen of the PingAccess setup wizard, then click **Finish**.
5. To customize and finalize the PingAccess setup, paste the URL you copied into your web browser and connect to the admin console of the instance you have just installed.

Start PingAccess

After you have installed PingAccess, you start the service so that you can complete your configuration.

About this task

Note: If you installed PingAccess using the Windows installer, the service is installed and started automatically.

Steps

1. In a command prompt or terminal window, change to the PingAccess `bin` directory:
   - On Linux: `cd` *`PA_HOME`*`/bin`
   - On Windows: `cd` *`PA_HOME`*`\bin`
2. Start the `run` script for the platform:
   - On Linux: `./run.sh`
   - On Windows: `run.bat`

Results

Wait for the script to execute. PingAccess is started when you see the message “PingAccess running...” in the command window.

Access the admin console for the first time

After you have installed and started PingAccess, you can access the admin console and perform configuration and first-time logon tasks.

Steps

1. Launch your browser and go to: https://<DNS_NAME>:9000

   <DNS_NAME> is the fully-qualified name of the machine running PingAccess.

   Info: If you have not yet installed a PingAccess license, the server will redirect you to the **License Upload** screen outside of the main UI. See the *PingAccess User Interface Reference Guide* for more information.
2. Sign in with the default username and password:

   Username: `Administrator`

   Password: `2Access`
3. Read and accept the license agreement.
4. Change the default administrator password on the **First Time Login** page, then click **Continue**.

   Info: The new password must conform to the rules specified by the `pa.admin.user.password.regex` property in `run.properties`.

   The PingAccess administrative console appears.

Results

Upon a successful login, PingAccess creates a backup of the current configuration to allow the administrator to revert any changes made. This backup is stored in *`PA_HOME`*`/data/archive`. The number of backup files can be controlled using the `pa.backup.filesToKeep` property in `run.properties`.

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

Access the PingAccess administrative API

You can access the PingAccess administrative API.

Steps

Send HTTP request to URL `https://<host>:<admin-port>/pa-admin-api/v3/<api-endpoint>`
.

Note: You must provide appropriate administrator credentials in the request.

Example

For example, the following cURL command will return a list of all defined applications by sending a Get request to the `applications` resource:

```
curl -k -u Administrator:Password1 -H "X-Xsrf-Header: PingAccess" https://
localhost:9000/pa-admin-api/v3/applications
```

- The `-u Administrator:Password1` parameter sends Basic Authentication header with the username `Administrator` and password `Password1`
- The `-k` parameter specifies to ignore HTTPS certificate issues
- The `-H "X-Xsrf-Header: PingAccess"` parameter sends an `X-XSRF-Header` with value `PingAccess`

Access the interactive administrative API documentation

You can view interactive documentation for the administrative API endpoints.

Steps

1. Launch your browser and go to URL `https://<host>:<admin-port>/pa-admin-api/v3/api-docs/`. For example, `https://localhost:9000/pa-admin-api/v3/api-docs/`.
2. The browser may prompt for credentials. Enter the administrator username and password.
3. Use the administrative API to perform a variety of administrative tasks, including those to gather information as seen in the following example that uses the interactive Administrative API documentation to see all defined applications:
   a. Click on the `/applications` endpoint to expand it.
   b. Click on the `GET` method (`GET /applications`) to expand it.
   c. Enter parameters values or leave all blank.
   d. Click **Try It Out**.
   e. The Request URL, Response Body, Response Code, and Response Headers appear.

Change configuration database passwords

You can change the file and user passwords for the PingAccess configuration database.

About this task

The PingAccess configuration database is protected by two passwords - a file password and a user password. These passwords both default to `2Access`, but should be changed for production environments.

Changing either password requires PingAccess be shut down.

Steps

1. Open a terminal window and change to the *<PA_HOME>*`/bin` directory.
2. Ensure that the `JAVA_HOME` environment variable is set correctly by executing the command `echo $JAVA_HOME`.
3. Ensure that the proper Oracle Java executable is in your path. Enter the command `java -version`. If this command returns a value indicating that the Java executable is not a supported version of Oracle Java, correct this issue before continuing.
4. Shut down PingAccess.
5. Optional: To change the database file password:
   a. If you are using Windows, run this command and note the output: `dbfilepasswd.bat` *`old_password new_password`*
   b. If you are using Linux, run this command and note the output: `./dbfilepasswd.sh` *`old_password new_password`*
   c. On all operating systems, update the `pa.jdbc.filepassword` property in *PA_HOME*`/conf/run.properties` with the obfuscated password output from the command given above.
6. Optional: To change the database user password:
   a. If you are using Windows, run this command and note the output: `dbuserpasswd.bat` *`file_password old_password new_password`*
   b. If you are using Linux, run this command and note the output: `./dbuserpasswd.sh` *`file_password old_password new_password`*
   c. On all operating systems, update the `pa.jdbc.password` property in *PA_HOME*`/conf/run.properties` with the obfuscated password output from the command given above.
7. Restart PingAccess.

Stop PingAccess

You can stop PingAccess as a prerequisite for some maintenance or uninstallation tasks.

Steps

1. Press `Ctrl+C` in the command-prompt or terminal window.
2. If PingAccess is running on Windows, press `y` when prompted to terminate the script.

Run PingAccess as a service

PingAccess can run as a service on Linux and Windows 64-bit operating systems, enabling PingAccess to start automatically when the operating system is started.

The service runs as the `root` (Linux) and `System` (Windows) user by default.

The following tasks allow you to manage PingAccess as a service:

- *Configure PingAccess to run as a Linux systemv service* on page 36
- *Configure PingAccess to run as a Linux systemd service* on page 36
- *Configure Multiple Instances of PingAccess as Linux services* on page 37
- *Remove the PingAccess Linux service* on page 37
- *Configure PingAccess to run as a Windows service* on page 38
- *Remove the PingAccess Windows service* on page 38

ⓘ **Tip:** Before performing the following procedures, ensure that PingAccess runs normally by manually starting the server. See *Run PingAccess for the First Time* for more infomation.

Configure PingAccess to run as a Linux systemv service

You can configure PingAccess to run as a Linux systemv service, causing it to start automatically when Linux starts.

About this task

> ⓘ **Note:** The service script will only start if JAVA_HOME and PA_HOME are set and the PingAccess license file is found.

Steps

1. Copy the PingAccess script file from `PA_HOME/sbin/linux/pingaccess` to `/etc/init.d`.
2. Optional: Create a new user to run PingAccess.
3. Create the folder `/var/run/pingaccess` and ensure that the user who will run the service has read and write permission to the folder.
4. Edit the script file `/etc/init.d/pingaccess` and set the values of following variables at the beginning of the script:
   - `export JAVA_HOME=` specify the Java install folder
   - `export PA_HOME=` specify the PingAccess install folder
   - `export USER=` (optional) specify user name to run the service, or leave empty for default
5. Register the service by running the command `chkconfig --add pingaccess` from the `/etc/init.d` folder.
6. Make the service script executable by running the command `chmod +x pingaccess`.

Results

Once registered, you can use the `service` command to control the pingaccess service. The available commands are:

start

Start the PingAccess Service

stop

Stop the PingAccess Service

restart

Restart the PingAccess Service

status

Show the status of the PingAccess service and the service PID

Example

The command `service pingaccess status` displays the current status of the running PingAccess service.

Configure PingAccess to run as a Linux systemd service

You can configure PingAccess to run as a Linux systemd service, causing it to start automatically when Linux starts.

About this task

ⓘ **Note:** The service script will only start if JAVA_HOME and PA_HOME are set and the PingAccess license file is found.

Steps

1. Copy the configuration file from `PA_HOME/sbin/linux/pingaccess.service` to `/etc/systemd/system/pingaccess.service`.
2. In the `pingaccess.service` file, replace the following variables:
   - `PA_HOME`
   - `PA_USER`
   - `JAVA_HOME`
3. Allow read/write activity on the service using the command `chmod 644 /etc/systemd/system/pingaccess.service`.
4. Load the systemd service using the command `systemctl daemon-reload`.
5. Enable the service using the command `systemctl enable pingaccess.service`.
6. Start the service using the command `systemctl start pingaccess.service`.

Configure Multiple Instances of PingAccess as Linux services

You can configure multiple instances of PingAccess on a single host as Linux services.

About this task

For hosts running multiple instances of PingAccess that need to be started as a service, follow the procedure used for *Configuring PingAccess as a Linux Service*, but make the following modifications to the script for each service:

Steps

- Use a unique script name for each instance
- Use a separate directory structure for each instance in the filesystem
- Configure the following settings in the script file for each instance:
  - `APPNAME`: A unique value for each instance
  - `PA_HOME`: The path to the PingAccess instance
  - `JAVA_HOME`: The path to the Java installation folder
  - `USER`: Optional value for the user name used to run the service

Remove the PingAccess Linux service

You can remove the PingAccess service from a Linux system.

About this task

ⓘ **Note:** The following commands must be run as the `root` user.

Steps

1. Stop the service by running `/etc/init.d/pingaccess stop`.
2. Run `chkconfig --delete pingaccess`.
3. Optional: Delete the `/etc/init.d/pingaccess` script.

Configure PingAccess to run as a Windows service

You can configure PingAccess to run as a Windows service, causing it to start automatically when Windows starts.

About this task

ⓘ **Info:** Before performing this procedure, ensure that PingAccess runs normally by manually starting the server (see *Run PingAccess for the First Time*).

ⓘ **Note:** If you installed PingAccess using the Windows installer, the service is installed and started automatically.

Steps

1. *Install PingAccess*.
2. Ensure you are logged in with full Administrator privileges.
3. Start a command prompt as an Administrator.
4. In the command prompt, change to the `PA_HOME\sbin\windows` directory and run the `install-service.bat` script.
5. Open the Windows **Control Panel# Administrative Tools# Services**.
6. Right-click **PingAccess Service** from the list of available services and select **Start**.
   The service starts immediately and restarts automatically on reboot. (You can change the default **Start type** setting in the **Properties** dialog.)

Remove the PingAccess Windows service

You can remove the PingAccess service from a Windows system.

About this task

To remove the PingAccess Windows Service, perform the following steps as an Administrator:

Steps

1. Open a command prompt.
2. Change the current directory to `PA_HOME\sbin\windows`.
3. Run `uninstall-service.bat`.
4. When the script has finished, remove the PA_HOME environment variable from the system.

Uninstall PingAccess

You can uninstall PingAccess.

Steps

1. *Stop PingAccess* on page 35.
2. Delete the PingAccess installation directory.

Configuring and Customizing PingAccess

Configure session management

Configure session management

This document provides information regarding session management using PingAccess. Use this document to learn about the concepts involved and to configure PingAccess for server-side session management using PingFederate.

Web Sessions

Web Sessions define the policy for Web application session creation, lifetime, timeouts, and their scope. Multiple Web Sessions may be configured to scope the session to meet the needs of a target set of applications. This improves the security model of the session by preventing unrelated applications from impersonating the end user. Use the following tasks to configure secure Web Sessions for use with specific applications and to configure global Web Session settings.

Application scoped Web Sessions

PingAccess Tokens can be configured to have their Web Sessions scoped to a specific application. This improves the security model of the session by preventing unrelated applications from impersonating the end user.

Several controls exist to scope the PA Token to an application:

Audience Attribute

The audience attribute defines who the token is applicable to and is represented as a short, unique identifier. Requests are rejected that contain a PA Token with an audience that differs from what is configured in the Web Session associated with the target Resource.

Audience Suffix

The audience attribute is also used as a suffix of the cookie name to ensure uniqueness. For example, PA.businessAppAudience.

Cookie Domain

The cookie domain can also optionally be set to limit where the PA Token is sent.

Info: In addition to these controls, parameters such as session timeout can be adjusted to match the policy requirements of each application.

Corresponding OAuth clients must be defined in PingFederate for each Web Session. Redirect URL whitelists defined in PingFederate dictate from which servers and domains the session can originate. Controlling this within PingFederate enables flexibility of the attribute contract (and its fulfillment) for that particular application. This ensures that each application and its associated policies only deal with attributes related to it.

Configure server-side session management

You can implement server-side session management in one of two ways.

- PingAccess can reject a PingAccess cookie associated with a PingFederate session that has been invalidated as a result of an end-user driven logout.
- The end user can initiate a logout from all PingAccess issued web sessions using a centralized logout.

The first of these scenarios provides increased scalability and security, ensuring the PingFederate session is terminated and that subsequent session validation requests are rejected. This scenario implies a user logout from PingAccess protected resources through the invalidation of the related PingFederate session.

The second scenario provides improved performance and end user experience. When the user explicitly logs out of the PingAccess issued session, all related PingAccess cookies are deleted, ensuring the client is no longer authenticated to resources protected by PingAccess. In this scenario, the user has explicitly logged out from all of those protected services.

PingAccess needs to be configured only for the first of these two scenarios. These options are not mutually exclusive, and can be combined to provide comprehensive session management at the server.

Configure PingFederate for session management
You can configure PingFederate to be able to revoke PingAccess session cookies.

Steps

1. Log in to the PingFederate Administrative Console
2. Navigate to **Server Configuration**# **Server Settings**# **Roles & Protocols**
3. Ensure that **Enable OAuth 2.0 Authorization Server (AS) role** and **OpenID Connect** are enabled. Create or modify an existing client.
4. From the main administration page, navigate to **OAuth Settings**# **Authorization Server Settings** and configure the authorization server settings.
5. Return to the main administration page, then go to **OAuth Settings**# **Client Management**
6. Create or modify an existing client.
7. Ensure that **Client Secret** is enabled, then enter a client secret to be used by PingAccess for authentication.
8. In the OpenID Connect section of the client's configuration page, enable **Grant Access to Session Revocation API**.

> **Note:** This setting is the main setting that enables the server-side session management feature in PingFederate.

9. Click **Save** to save your changes.

Configure PingFederate for user-initiated single logout
You can configure PingFederate to provide PingAccess with access to the PingFederate-managed session.

Steps

1. Log in to the PingFederate administrative console.
2. Navigate to **OAuth Settings**# **Authorization Server Settings**.
3. Select **Track User Sessions for Logout** under **OpenID Connect Settings**.
4. Click **Save**.
5. Navigate to **OAuth Settings**# **OpenID Connect Policy Management** and click an existing policy.
6. Click **Manage Policy**, then enable **Include Session Identifier in ID Token**.

   For more information about configuring an OpenID Connect Policy, see *Configuring OpenID Connect Policies* in the *PingFederate Administrator's Manual*.
7. Click **Save**.
8. Navigate to **OAuth Settings**# **Client Management** and select the client to be used by PingAccess.

9. In the OpenID Connect section of the client's configuration page, enable **PingAccess Logout Capable**.

> **Tip:** If this option is not available, ensure that the **Track User Sessions for Logout** setting change made in step 3 was saved.

10. Click **Save**.

Configure PingAccess for server-side session management
You can configure PingAccess to enable server-side session management.

Steps

1. Log in to the PingAccess administrative console.
2. Click **Settings# Access# Web Sessions**.
3. Either *create a new web session* or *edit an existing web session*.
4. In the **Client ID** field, enter the Client ID defined in PingFederate.
5. Enter the client secret associated with the specified Client ID.
6. Click the **Advanced** tab.
7. Select **Validate Session** to enable the server-side session management feature.
8. Click **Save**.

Configure logging

Configure logging

This document describes the types of logging performed by PingAccess and provides instructions for configuring PingAccess logging.

Security audit logging

The PingAccess audit logs record a selected subset of transaction log information at runtime plus additional details, intended to facilitate security auditing and regulatory compliance.

The logs are located in *`PA_HOME`*`/log/`. Elements recorded in these logs are described in the table below, and are configured in `conf/log4j2.xml`.

> **Important:** Log files can be viewed or modified using a variety of common applications. As such, it is possible for log files to be manipulated to include untrusted or malicious data. Administrators should take appropriate steps to secure these files. Furthermore, these files should not be opened in applications that could allow for data execution, such as internet browsers or Microsoft Office products. It is highly recommended that administrators open these files in a common, lightweight text editor.

PingAccess generates these audit logs:

`pingaccess_engine_audit.log`

Records transactions of configured Resources. Additionally, the log records transaction details when PingAccess sends requests to PingFederate (for example, STS, OAuth2, JWS).

`pingaccess_api_audit.log`

Records PingAccess administrative API transactions. These transactions represent activity in the PingAccess administrative console. This log also records transaction activity if you are using scripts to configure PingAccess.

`pingaccess_agent_audit.log`

Records transactions between PingAccess Agents and the PingAccess Engine.

Audit log configuration

Item	Description
`%d`	Transaction time.
`exchangeId`	Identifies the ID for a specific request/response pair.
`AUDIT.applicationID`	Specifies the ID of the requested application.
`AUDIT.applicationName`	Specifies the name of the requested application.
`AUDIT.resourceID`	Specifies the ID of the requested resource.
`AUDIT.resourceName`	Specifies the name of the requested resource.
`AUDIT.pathPrefix`	Specifies the path prefix of the requested application or resource.
`AUDIT.pathPrefixType`	Indicates the pattern type of the path prefix, `Wildcard` or `Regex`.
`AUDIT.authMech`	Mechanism used for authentication. Engine Auditing - Cookie (WAM session), OAuth, unknown (for example, pass-through or static assets). Pass-through assets are Resources with no policies or Web session configured. Admin Auditing - Basic, OAuth, Cookie, unknown (unknown displays only in an authentication failure).
`AUDIT.client`	IP address of the requesting client.
`AUDIT.failedRuleName`	Name of the Rule that failed. If no Rule failure occurred, this field is blank. This element is applicable only to the pingaccess_engine_audit.log.
`AUDIT.failedRuleType`	Type of Rule that failed. If no Rule failure occurred, this field is blank. This element is applicable only to the pingaccess_engine_audit.log.
`AUDIT.failedRuleClass`	The Java class of Rule that failed. If no Rule failure occurred, this field is blank. This element is applicable only to the pingaccess_engine_audit.log.
`AUDIT.failedRuleSetName`	Name of the containing Rule Set that failed. If no Rule failure occurred, this field is blank. This element is applicable only to the pingaccess_engine_audit.log.
`AUDIT.host`	PingAccess host name or IP address.
`AUDIT.targetHost`	Backend target that processed the request and generated a response to the PingAccess engine.
`AUDIT.method`	HTTP method of the request. For example, GET.
`AUDIT.resource`	Name of the Resource used to fulfill the request. This element is applicable only to the pingaccess_engine_audit.log.
`AUDIT.responseCode`	HTTP status code of the response. For example, 200.

Item	**Description**
`AUDIT.requestUri`	Request URI portion of the request (for example, /foo/bar).
`AUDIT.subject`	Subject of the transaction.
`AUDIT.trackingId`	The PingFederate Tracking ID. This element can be used to help correlate audit information in the PingAccess audit log with information recorded in the PingFederate audit log. The value of this depends on whether the application type is `Web` or `API`. If the application type is `Web`, the value is presented as `tid:`*`<Session_Identifier>`*. The <*Session_Identifier*> can be used by the *PingFederate Session Revocation API* to revoke the session without disabling the user in the identity store. If the application type is `API`, the value is presented as `atid:`*`<Hash>`* . The <*Hash*> value is derived from the OAuth Access token for the session, and only serves as an identifier; it cannot be used for session revocation.
`AUDIT.reqReceivedMillisec`	Time in milliseconds (since 1970) that a client request was first received
`AUDIT.reqSentMillisec`	Time in milliseconds (since 1970) that the agent or engine sent a backchannel or proxy request
`AUDIT.respReceivedMillisec`	Time in milliseconds (since 1970) that the agent or engine received a response from a backchannel call or proxy request
`AUDIT.respSentMillisec`	Time in milliseconds (since 1970) that a response was sent back to the client
`AUDIT.roundTripMS`	The respSentMillisec time minus the reqReceivedMillisec time. This represents the total number of milliseconds it took PingAccess to respond to a client's request (including the proxyRoundTripMS).
`AUDIT.proxyRoundTripMS`	The respReceivedMillisec time minus the reqSentMillisec time. This represents the total number of milliseconds PingAccess was waiting for another entity to respond to a backchannel call or proxy request.
`AUDIT.userInfoReqSentMillisec`	The time in milliseconds (since 1970) that the engine sent a request to the token provider's OIDC UserInfo endpoint.
`AUDIT.userInfoRespReceivedMillisec`	The time in milliseconds (since 1970) that the engine received a response from the token provider's OIDC UserInfo endpoint.

Item	Description
`AUDIT.userInfoRoundTripMS`	The userInfoRespReceivedMillisec minus the userInfoReqSentMillisec time. This represents the total number of milliseconds it took PingAccess to receive a response from the token provider's UserInfo endpoint.
`appRequestHeader{a-header-name}`	HTTP request header value for the given HTTP request header name. Represents the header value that PingAccess sends to the back end site.
`appResponseHeader{a-header-name}`	HTTP response header value for the given HTTP request header name. Represents the header value received from the application.
`clientRequestHeader{a-header-name}`	HTTP request header value for the given HTTP request header name. Represents the header value received from the client.
`clientResponseHeader{a-header-name}`	HTTP response header value for the given HTTP request header name. Represents the header value returned to the client.

Logging

PingAccess logging is handled by the log4j2 asynchronous logging library, configured using `conf/log4j2.xml`.

Info: Audit logs are also configurable in `conf/log4j2.xml`. These logs record a selected subset of transaction log information at runtime plus additional details (see *Security Audit Logging*).

By default, logging information is output to *`PA_HOME`*`/logs/pingaccess.log`, and file logging uses the *rolling* file appender. PingAccess keeps a maximum of 10 log files, each with a maximum size of 100 MB. Once 10 files accumulate, PingAccess deletes the oldest. These defaults can be changed by locating and modifying the following properties in the `<Appenders>` section of `log4j2.xml`:

- Changing the log file name:

```
<RollingFile name="File"
            fileName="${sys:pa.home}/log/pingaccess.log"
            filePattern="${sys:pa.home}/log/pingaccess.log.%i"
            ignoreExceptions="false">
```

- Setting the maximum log size:

```
<SizeBasedTriggeringPolicy size="100000 KB"/>
```

- Setting the maximum number of log files:

```
<DefaultRolloverStrategy max="10"/>
```

In addition to the standard log4j2 items, PingAccess adds the following custom item that can be used in the `log4j2.xml <PatternLayout>` configuration:

Item	Description
`exchangeId`	Identifies the ID for a specific request/response pair.

For example, the following line from `log4j2.xml` incorporates the `exchangeId` in the output:

```
<pattern>%d{ISO8601} %5p [%X{exchangeId}] %c:%L - %m%n</pattern>
```

ⓘ **Note:** The `%X` conversion character is required for the `exchangeId` to be displayed properly.

Configure log levels

You can define log levels for specific package or class names in order to get more (or less) logging from a class or group of classes.

About this task

If the log level is not specified for a particular package or class, the settings for the root logger are inherited.

Steps

1. Open `conf/log4j2.xml` in an editor.
2. Locate the `<AsyncLogger>` element for the package or class you wish to adjust the logging level for.

   ```
   <AsyncLogger name="com.pingidentity" level="DEBUG" additivity="false"
    includeLocation="false">
   ```
3. Modify the `level` attribute to set the desired log level.

   Valid values are `OFF`, `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`.
4. Save the modified file. PingAccess will automatically make the changes effective within 30 seconds.

Configure a class or package log level

You can use the log4j2.xml file to configure the log level for a class or package.

Steps

1. Open `conf/log4j2.xml` in an editor. Class or package loggers are defined in the `<AsyncLogger>` `name` attribute. For example, cookie logging is enabled using the line:

   ```
   <AsyncLogger name="com.pingidentity.pa.core.interceptor.CookieLoggingInterceptor"
    level="TRACE" additivity="false" includeLocation="false">
      <AppenderRef ref="File"/>
   </AsyncLogger>
   ```
2. Set the `level` value in the `<AsyncLogger>` element to one of the following values:

   `OFF`, `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`.

   For example, to apply `TRACE` level logging for the `com.pingidentity` package, locate the following line:

   ```
   <AsyncLogger name="com.pingidentity" level="DEBUG" additivity="false"
    includeLocation="false">
   ```

   and change it to:

   ```
   <AsyncLogger name="com.pingidentity" level="TRACE" additivity="false"
    includeLocation="false">
   ```
3. Save the file.

Enable cookie logging

Cookie logging is an optional feature in the TRACE log level.

Steps

1. Edit `conf/log4j2.xml` and uncomment the following section:

```
<AsyncLogger
 name="com.pingidentity.pa.core.interceptor.CookieLoggingInterceptor"
 level="TRACE" additivity="false" includeLocation="false">
     <AppenderRef ref="File"/>
</AsyncLogger>
```

2. Save the file.

Append log messages to syslog and the console

You can enable additional output destinations (called *appenders*).

About this task

Console and syslog appenders are pre-configured in `log4j2.xml`, but are disabled by default.

To enable additional appenders, perform the following steps:

Steps

1. Open `conf/log4j2.xml` in an editor.
2. Locate the following lines in the `<Loggers>` element:

```
<AsyncLogger name="com.pingidentity" level="DEBUG" additivity="false"
 includeLocation="false">
    <AppenderRef ref="File"/>
    <!--<AppenderRef ref="CONSOLE" />-->
    <!--<AppenderRef ref="SYSLOG" />-->
</AsyncLogger>
```

Note: If you have customized logging to enable logging for additional classes, locate the `<AsyncLogger>` element that is relevant to the class in question. This class is defined in the `<AsyncLogger>` `name` attribute.

3. Uncomment the `<AppenderRef>` element that applies to the appender you wish to enable.

Note: PingAccess will rescan the logging configuration within 30 seconds and make the change active automatically.

4. Save the file.

Write logs to other formats

You can write the audit logs to an *Oracle or SQL Server database*.

You may also configure PingAccess to write the audit logs to a differently formatted log file that can easily be digested by *Splunk*.

Write logs to databases

You can enable database logging for the API, engine, and agent audit logs in `conf/log4j2.db.properties`.

About this task

Scripts are provided in `conf/log4j/sql-scripts` to create the necessary tables. PingAccess supports logging to Oracle, SQL Server, and PostgreSQL databases.

Steps

1. Ensure that your database driver JAR file is installed in the *`PA_HOME`*`/lib` directory. Restart PingAccess after installing the driver.
2. In `conf/log4j2.xml`, uncomment one or more of the preset appender configurations listed below:
   - **Oracle**
     - **For Administrative API audit logging**: Uncomment the `<JDBC>` element with the `name="ApiAuditLog-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Engine audit logging**: Uncomment the `<JDBC>` element with the `name="EngineAuditLog-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Agent audit logging**: Uncomment the `<JDBC>` element with the `name="AgentAuditLog-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
   - **SQL Server**
     - **For Administrative API audit logging**: Uncomment the `<JDBC>` element with the `name="ApiAuditLog-SQLServer-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Engine audit logging**: Uncomment the `<JDBC>` element with the `name="EngineAuditLog-SQLServer-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Agent audit logging**: Uncomment the `<JDBC>` element with the `name="AgentAuditLog-SQLServer-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
   - **PostgreSQL**
     - **For Administrative API audit logging**: Uncomment the `<JDBC>` element with the `name="ApiAuditLog-PostgreSQL-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Engine audit logging**: Uncomment the `<JDBC>` element with the `name="EngineAuditLog-PostgreSQL-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.
     - **For Agent audit logging**: Uncomment the `<JDBC>` element with the `name="AgentAuditLog-PostgreSQL-Database"` attribute specified, along with the following `<RollingFile>` and `<PingAccessFailover>` elements.

   Note: The `<PingAccessFailover>` element is used to define how PingAccess logging fails over if a connection to the primary database is not accessible. Use the `retryIntervalSeconds` attribute to specify the number of seconds that must pass before retrying the primary JDBC appender.

3. In `conf/log4j2.db.properties`, replace the placeholder parameter values for each enabled appender with valid values to provide access to the database.

   Info: You can obfuscate the password used to access the database by running either **`obfuscate.sh`** or **`obfuscate.bat`**, located in *`PA_HOME`*`/bin`. Use the database password as an argument, then copy the output into the password configuration property for the appender in *`PA_HOME`*`/conf/log4j2.db.properties`.

4. In conf/log4j2.xml, uncomment the `<AppenderRef>` elements in each respective `<Logger>` section as shown in the following examples:

   Oracle:

```
<!-- Audit Log Configuration-->
<Logger name="apiaudit" level="INFO" additivity="false">
    <AppenderRef ref="APIAuditLog-File"/>
    <AppenderRef ref="ApiAuditLog-Database-Failover"/>
    <!--<AppenderRef ref="ApiAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="ApiAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="ApiAudit2Splunk"/>-->
</Logger>
<Logger name="engineaudit" level="INFO" additivity="false">
    <AppenderRef ref="EngineAuditLog-File"/>
    <AppenderRef ref="EngineAuditLog-Database-Failover"/>
    <!--<AppenderRef ref="EngineAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="EngineAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="EngineAudit2Splunk"/>-->
</Logger>
<Logger name="agentaudit" level="INFO" additivity="false">
    <AppenderRef ref="AgentAuditLog-File"/>
    <AppenderRef ref="AgentAuditLog-Database-Failover"/>
    <!--<AppenderRef ref="AgentAuditLog-SQLServer-Database-Failover"/>-->
    <!--<AppenderRef ref="AgentAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="AgentAudit2Splunk"/>-->
</Logger>
```

   SQL Server

```
<!-- Audit Log Configuration-->
<Logger name="apiaudit" level="INFO" additivity="false">
    <AppenderRef ref="APIAuditLog-File"/>
    <!--<AppenderRef ref="ApiAuditLog-Database-Failover"/>-->
    <AppenderRef ref="ApiAuditLog-SQLServer-Database-Failover"/>
    <!--<AppenderRef ref="ApiAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="ApiAudit2Splunk"/>-->
</Logger>
<Logger name="engineaudit" level="INFO" additivity="false">
    <AppenderRef ref="EngineAuditLog-File"/>
    <!--<AppenderRef ref="EngineAuditLog-Database-Failover"/>-->
    <AppenderRef ref="EngineAuditLog-SQLServer-Database-Failover"/>
    <!--<AppenderRef ref="EngineAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="EngineAudit2Splunk"/>-->
</Logger>
<Logger name="agentaudit" level="INFO" additivity="false">
    <AppenderRef ref="AgentAuditLog-File"/>
    <!--<AppenderRef ref="AgentAuditLog-Database-Failover"/>-->
    <AppenderRef ref="AgentAuditLog-SQLServer-Database-Failover"/>
    <!--<AppenderRef ref="AgentAuditLog-PostgreSQL"/>-->
    <!--<AppenderRef ref="AgentAudit2Splunk"/>-->
</Logger>
```

   PostgreSQL

```
<!-- Audit Log Configuration-->
<Logger name="apiaudit" level="INFO" additivity="false">
    <AppenderRef ref="APIAuditLog-File"/>
    <!--<AppenderRef ref="ApiAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="ApiAuditLog-SQLServer-Database-Failover"/>-->
    <AppenderRef ref="ApiAuditLog-PostgreSQL"/>
    <!--<AppenderRef ref="ApiAudit2Splunk"/>-->
```

```
</Logger>
<Logger name="engineaudit" level="INFO" additivity="false">
    <AppenderRef ref="EngineAuditLog-File"/>
    <!--<AppenderRef ref="EngineAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="EngineAuditLog-SQLServer-Database-Failover"/>-->
    <AppenderRef ref="EngineAuditLog-PostgreSQL"/>
    <!--<AppenderRef ref="EngineAudit2Splunk"/>-->
</Logger>
<Logger name="agentaudit" level="INFO" additivity="false">
    <AppenderRef ref="AgentAuditLog-File"/>
    <!--<AppenderRef ref="AgentAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="AgentAuditLog-SQLServer-Database-Failover"/>-->
    <AppenderRef ref="AgentAuditLog-PostgreSQL"/>
    <!--<AppenderRef ref="AgentAudit2Splunk"/>-->
</Logger>
```

5. Create the database tables. Scripts to create database tables are located in `conf/log4j/sql-scripts`.

Note: The scripts are written to handle the default list of elements for the relevant database log appender. Any changes to the list require corresponding changes to the SQL table creation script, or to the table itself if it already exists. For more information on working with these scripts, see Oracle, PostgreSQL, or MS SQL Server documentation.

Important: For PostgreSQL database scripts, use of the default **public** schema is not recommended. To run the scripts against a different schema, choose one of the following options:

- Prepend the schema before the table name. For example, `api_audit_log` would become `my_schema.api_audit_log`
- Run the script via **psql** and specify an options parameter to define the schema. For example:

```
psql postgresql://<user>@<db_hostname>:5432/<db_name>?options=--
search_path=<schema> -f api-audit-log-postgresql.sql
```

Write audit logs for Splunk

You can configure PingAccess to write audit logs to a format for Splunk.

About this task

Splunk is enterprise software that allows for monitoring, reporting, and analyzing consolidated log files. Splunk captures and indexes real-time data into a single searchable repository from which reports, graphs, and other data visualization can be generated.

Steps

1. In `conf/log4j2.xml`, uncomment the `<RollingFile>` and `<AppenderRef>` elements for the Splunk appenders you wish to enable:

```
<!--
<RollingFile name="ApiAudit2Splunk"
             fileName="${sys:pa.home}/log/pingaccess_api_audit_splunk.log"
             filePattern="${sys:pa.home}/log/pingaccess_api_audit_splunk.
%d{yyyy-MM-dd}.log"
             ignoreExceptions="false">
    <PatternLayout>
        <pattern>%d{ISO8601} exchangeId="%X{exchangeId}"
 trackingId="%X{AUDIT.trackingId}" subject="%X{AUDIT.subject}"
 authMech="%X{AUDIT.auth
Mech}" client="%X{AUDIT.client}" method="%X{AUDIT.method}"
 requestUri="%X{AUDIT.requestUri}" responseCode="%X{AUDIT.responseCode}"
 %n</pattern>
    </PatternLayout>
    <Policies>
        <TimeBasedTriggeringPolicy />
    </Policies>
</RollingFile>
-->
```

Note: `<RollingFile>` elements are also present for `EngineAudit2Splunk` and `AgentAudit2Splunk`. They are structured similarly to the `ApiAudit2Splunk` example shown above.

```
<Logger name="apiaudit" level="INFO" additivity="false">
    <AppenderRef ref="APIAuditLog-File"/>
    <!--<AppenderRef ref="ApiAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="ApiAudit2Splunk"/>-->
</Logger>
<Logger name="engineaudit" level="INFO" additivity="false">
    <AppenderRef ref="EngineAuditLog-File"/>
    <!--<AppenderRef ref="EngineAuditLog-Database-Failover"/>-->
    <!--<AppenderRef ref="EngineAudit2Splunk"/>-->
</Logger>
<Logger name="agentaudit" level="INFO" additivity="false">
    <AppenderRef ref="AgentAuditLog-File"/>
    <!--<AppenderRef ref="AgentAuditLog-Database-Failover"/>-->
  <!--<AppenderRef ref="AgentAudit2Splunk"/>-->
</Logger>
```

2. Save the file.

Note: PingAccess automatically updates its configuration within 30 seconds; a restart is not required for this change to be effective.

3. Download and install the Splunk Universal Forwarder on the machine running PingAccess.
4. Configure the Universal Forwarder to monitor `logs/pingaccess_api_audit_splunk.log`, `logs/pingaccess_agent_audit_splunk.log`, or `logs/pingaccess_engine_audit_splunk.log`.

Info: For detailed installation and configuration instructions, consult the Splunk documentation accompanying the Universal Forwarder.

Customize and localize PingAccess

Customization of user-facing pages

PingAccess supplies templates to provide information to the end user. These template pages use the Velocity template engine, an open-source Apache project, and are located in the *PA_HOME*`/conf/template` directory.

You can modify most of these pages in a text editor to suit the particular branding and informational needs of your PingAccess installation. (Cascading style sheets and images for these pages are included in the *PA_HOME*`/conf/static/pa/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 PingAccess server supplies values for the Velocity variables used in the template.

Important: If you have modified the reserved application context root using the PingAccess Admin API, file system requests to the configured reserved application context root will be translated to `/pa`. This allows the file system behavior for PingAccess resources to remain unchanged. Thus, if the reserved context root is set to `/ping`, templates and other resources would still be stored on the file system in the `/pa` directory, as indicated by this document.

For information about Velocity, refer to the *Velocity project documentation* on the Apache Web site. Changing Velocity or JavaScript code is not recommended. The following variables are the only variables that can be used for rendering the associated Web-browser page.

Variable	Description
`title`	The browser tab title for the message. For example, Not Found.
`header`	The header for the message. For example, Not Found.
`info`	The information for the message. For example, No Resource configured for request.
`exchangeId`	A value that identifies the request/response pair. This can be used to locate messages in the PingAccess logs.
`trackingId`	A value that identifies either the tracking ID (identified with a `tid:` prefix) or an Access Token ID (identified with a `atid:` prefix). This can be used to identify the session in the PingAccess and PingFederate logs.

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 the table below). For example, if Rule evaluation fails, the user's browser is directed to the Policy error-handling page. The following table describes each template.

Template File Name	Purpose	Type	Action
`admin.error.page.template.html`	Indicates an error occurred while the admin console was processing a request	Error	Consult *`PA_HOME/`*`log/pingaccess.log` to determine the underlying cause of the issue.
`general.error.page.template.html`	Indicates that an unknown error has occurred and provides an error message.	Error	Consult *`PA_HOME/`*`log/pingaccess.log` to determine the underlying cause of the issue.
`general.loggedout.page.template.html`	Displayed when a user logs out of PingAccess.	Normal	User should close the browser.

Template File Name	Purpose	Type	Action
`oauth.error.json`	Indicates that Rule evaluation has failed and provides an optional error message. To customize this information, see Error-Handling Fields for OAuth Rules documentation.	Normal	If necessary, consult the audit logs in *PA_HOME*/`log` for details about why the policy denied the request.
`policy.error.page.template.html`	Indicates that Rule evaluation has failed and provides an optional error message. To customize this information, see Error-Handling Fields for Rules documentation.	Normal	If necessary, consult the audit logs in *PA_HOME*/`log` for details about why the policy denied the request.

Note: The templates stored in *PA_HOME*`/conf/template/system` are system templates, and should not be modified.

Localization of user-facing pages

In addition to the use of Velocity templates to change the look and feel of user-facing pages, administrators can provide localized versions of user-facing status messages generated by PingAccess.

In *PA_HOME*`/conf/localization/`, properties files contain the messages to be returned to the client in various languages; by default, only English language messages are provided, using the default `pa-messages.properties` file. This file serves as a fallback for any message not found in other files in the directory.

The selection of a messages file is determined based on several different factors:

- The browser's `Accept-Langauge` header, based on a best-match first check against the `pa-messages` files
- The value of a cookie named `ping-accept-language`, which can be defined by the protected application
- A custom-developed PingAccess add-on that can customize the order of localization resolution

The default behavior allows the `ping-accept-language` cookie to override the browser preferences, and if that cookie is not set, then to use the `Accept-Language` header preference order, starting with the highest priority preference and trying to match the locale exactly. If none of the specified locales cannot be matched exactly, a more generic locale will be used, starting with the highest priority value.

If no matches are found, then the value in the `pa-messages.properties` file is used.

For example, suppose your browser had the following `Accept-Language` header:

```
Accept-Language: fr-CA;q=0.9, en-US;q=0.8
```

and PingAccess attempted to display a localized version of the message for:

```
pa.response.status.service.unavailable
```

The order in which PingAccess searches for the string to display is:

1. `pa-messages_fr_CA.properties`
2. `pa-messages_en_US.properties`
3. `pa-messages_fr.properties`
4. `pa-messages_en.properties`
5. `pa-messages.properties`

If the `ping-accept-language` cookie is set by the protected application to the value `en-US`, then the above list would be ignored, and PingAccess would search for the string in:

1. `pa-messages_en_US.properties`
2. `pa-messages_en.properties`
3. `pa-messages.properties`

Important: Most browsers support the use of an ordered list of languages, however, Safari is an exception to this. Even though the system supports an ordered list of languages, only the preferred language is sent with its requests.

Upgrading PingAccess

Upgrade PingAccess

Upgrading your environment

You can upgrade your PingAccess deployment. The procedure varies depending on your environment.

- If you have a standalone PingAccess deployment not installed using the RHEL or Windows installer, use the *Upgrade a PingAccess standalone version* on page 53 procedure.
- If you have a PingAccess cluster, use the *Upgrade a PingAccess cluster* on page 55 procedure.
- If you installed PingAccess on RHEL using the installer, use the *Upgrade RHEL using the installer* on page 57 procedure.
- If you installed PingAccess on Windows using the installer, use the *Upgrade Windows using the installer* on page 56 procedure.

Upgrade a PingAccess standalone version

You can upgrade a standalone PingAccess deployment to a newer version.

Before you begin

- Create a backup of your existing PingAccess configuration. If the upgrade fails, you can restore your environment from this backup.
- Review the release notes for every version between your current version and the target version.

Important: In release 5.0, there are potentially breaking changes to the SDK for Java, Groovy scripts, and the Administrative API. For information on these changes and the actions administrators may need to take, review the *Upgrade considerations* on page 13 and the *PingAccess Release Notes for release 5.0*.

- Verify that you have the following:
  - The PingAccess Upgrade Utility archive
  - The PingAccess distribution `.zip` file
  - Your PingAccess license file
  - Login access to the PingAccess host, as the utility is run on the host
  - Administrator credentials for the running PingAccess instance
- Verify that Basic Authentication is configured and enabled for the running PingAccess instance.
- Verify that the PingAccess host is running.

ⓘ **Important:** If you have set `security.overridePropertiesFile=false` in `$JAVA_HOME/jre/lib/java.security`, the upgrade utility might fail, as the PingAccess Upgrade Utility uses an override to enable deprecated ciphers and protocols during the upgrade process.

About this task

Use the PingAccess Upgrade Utility to upgrade from PingAccess 3.0 or later (the source version) to the most recent version (the target version).

The upgrade utility starts an instance of PingAccess with an administrative listener on port 9001. This port number can be changed using the `run.bat` or `run.sh` -p parameter. This port configuration is only used for the upgrade. The default port is used by the upgraded server when the upgrade is complete.

Any warnings or errors encountered are recorded in `log/upgrade.log`, as well as on the screen while the utility is being run.

ⓘ **Important:**

If your installation uses custom plugins, they will need to be rebuilt against the new (5.0) SDK. You will then run the upgrade utility manually with the new -i command-line option to include their plugin directory. To migrate your custom plugins, see the *PingAccess Addon SDK for Java Migration Guide*.

ⓘ **Info:** During the upgrade, it is important to not make any changes to the running PingAccess environment.

Steps

1. Copy the upgrade utility `.zip` file to the PingAccess host and extract it.
2. Change to the upgrade utility's `bin` directory.

3. Run the PingAccess Upgrade Utility:
   - On Windows: `run.bat [-p <admin_port>] [-i <directory>] <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`
   - On Linux: `./run.sh [-p <admin_port>] [-i <directory>] <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`

   For example: `./run.sh -p 9002 -i MyJARDir pingaccess-4.3 pingaccess-5.0 pingaccess-5.0.zip pingaccess50.lic`

   Parameter definitions

   The command-line parameters are the same regardless of the platform, and are defined as follows:

Parameter	Value description
<admin_port>	Optional port to be used by the temporary PingAccess instance run during the upgrade. The default is 9001.
<directory>	A directory containing additional library JAR files (e.g. plugins, JDBC drivers) to be copied into the target installation
<sourcePingAccessRootDir>	The PA_HOME for the source PingAccess version
<outputDir>	The target directory which will contain the unpacked PingAccess distribution
<pingaccessZip>	The PingAccess distribution for the target version
<newPingAccessLicense>	The path to the PingAccess license file to use for the target version

Next steps

After you have completed the upgrade, *Perform post-upgrade tasks* on page 58.

Upgrade a PingAccess cluster

You can upgrade a PingAccess cluster to a newer version.

Before you begin

- Create a backup of your existing PingAccess configuration. If the upgrade fails, you can restore your environment from this backup.
- Review the release notes for every version between your current version and the target version.

> **Important:** In release 5.0, there are potentially breaking changes to the SDK for Java, Groovy scripts, and the Administrative API. For information on these changes and the actions administrators may need to take, review the *Upgrade considerations* on page 13 and the *PingAccess Release Notes for release 5.0*.

- Verify that each node is using the same PingAccess version. You can check the version by viewing the `<PA_HOME>/pingaccess-admin-ui-<version number>.jar` file.
- Verify that the PingAccess administrative node is running.
- Verify that basic authentication is configured and enabled for the running PingAccess administrative node.

About this task

To upgrade a cluster, perform the following steps:

Steps

1. On the administrative node, extract the upgrade utility `.zip` file.
2. Change to the upgrade utility's `bin` directory.
3. Run the PingAccess upgrade utility:
   - On Windows: `run.bat [-p <admin_port>] [-i <directory>] <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`
   - On Linux: `./run.sh [-p <admin_port>] [-i <directory>] <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`
4. Review the upgrade log. If it records any manual post-upgrade tasks, perform these steps:
   a. Open the `run.properties` file and change the upgraded administrative console's `admin.port` and `clusterconfig.port` values to a temporary value. This file is in the `<PA_HOME>/conf/` directory on Linux systems and the `<PA_HOME>\conf\` directory on Windows systems.
   b. Start the upgraded administrative console using the `<PA_HOME>/bin/run.sh` command on Linux systems or the `<PA_HOME>\bin\run.bat` command on Windows systems.
   c. Perform any manual post-upgrade tasks recorded in the upgrade log.
   d. Shut down the upgraded administrative console.
   e. Change the upgraded administrative console's `admin.port` and `clusterconfig.port` values back to the original values.
5. Run the upgrade utility on each engine node.
6. Shut down the entire cluster.
7. Start the upgraded administrative node.
8. Start each upgraded engine node.

Next steps

After you have completed the upgrade, *Perform post-upgrade tasks* on page 58.

Upgrade Windows using the installer

If you installed PingAccess using the Windows installer, use this procedure to upgrade using the installer.

Before you begin

- Review the *Upgrade considerations* on page 13.

About this task

Important: If additional JAR files (such as custom plugins and JDBC drivers) have been added to the existing PingAccess `/lib` directory, the 5.0-Beta installer cannot be used to perform the upgrade. Instead, run the upgrade utility manually, using the -i command-line option to specify the JAR files to be included.

Steps

1. Download the installer.
2. Start the installer. The existing installation is detected.
3. Choose **Yes** to upgrade the installation.

4. Choose a license file and specify a temporary admin port. Click **Next**.

   > **Note:** The temporary admin port is not required when upgrading a cluster node.

5. Specify the administrator credentials. Click **Next**.

   > **Note:** Administrator credentials are not required when upgrading a cluster node.

6. Click **Finish**.

Next steps

After you have completed the upgrade, *Perform post-upgrade tasks* on page 58.

Upgrade RHEL using the installer

If you installed PingAccess using the RHEL installer, use this procedure to upgrade using the installer.

Before you begin

- Review the *Upgrade considerations* on page 13.

About this task

> **Note:** On supported systems, you are given the option to upgrade your existing systemv scripting service to the newer systemd service. Upgrading the service will replace the existing service.

> **Important:** If additional JAR files (such as custom plugins and JDBC drivers) have been added to the existing PingAccess`/lib` directory, the 5.0-Beta installer cannot be used to perform the upgrade. Instead, the Upgrade Utility should be run manually, using the `-i` command-line option to specify the JAR files to be included.

Steps

1. Download the installer script.
2. Launch the script using the command **`sudo -E ./pa-install-<version>.sh -u`**.
3. Specify the instance you want to upgrade.

   > **Important:** If you are upgrading a PingAccess cluster, always upgrade the console first.

4. Confirm the selection.
5. Specify a license file.
6. Specify the temporary admin port.

   > **Note:** The temporary admin port is not required when upgrading a cluster node.

7. Specify whether you have additional plugin or JDBC driver JAR files to include during upgrade.

   > **Note:** This step is required when plugin JARs are present and you are upgrading to a version that is not SDK backward-compatible. For more information, see *Upgrade a PingAccess standalone version* on page 53 and the *PingAccess Addon SDK for Java Migration Guide*.

8. If you selected `yes` to include additional JAR files, enter the path to the `include` directory. The JAR files in this directory will be copied into the new PingAccess installation instead of migrating additional library JARs from the existing installation.

9. Provide the administrator credentials.

   Note: Administrator credentials are not required when upgrading a cluster node.

10. Allow the upgrade to complete.

Next steps

After you have completed the upgrade, *Perform post-upgrade tasks* on page 58.

Perform post-upgrade tasks

After you have upgraded your PingAccess deployment using the upgrade utility or the installer, you must perform several post-upgrade tasks to ensure that the target version works correctly.

About this task

To see details about the upgrade, examine `log/upgrade.log`. To see details about the migrated configuration data, examine `log/audit.log`.

After the upgrade utility or installer has completed, perform the following steps:

Steps

1. Review any warnings returned by the upgrade utility and take the actions indicated in the table below.

   At the end of an upgrade, the PingAccess Upgrade Utility or installer records any manual steps that require user intervention both in the command-line output and in `log/upgrade.log` at the WARN level. Information that does not require user intervention is added to the `log/upgrade.log` at the INFO level.
2. Generate new obfuscated passwords for the pa.jdbc.password, pa.jdbc.filepassword, and pa.keystore.pw parameters in `conf/run.properties`:
   - If you are on a Linux host, run this command:

     ```
     obfuscafe.sh password
     ```
   - If you are on a Windows host, run this command:

     ```
     obfuscate.bat password
     ```
3. Copy the obfuscated password and paste it into the parameter in `run.properties` that corresponds to the password being re-obfuscated.
4. Review the HTTP requests configuration to ensure the use of the IP source settings is appropriate for the environment. During the upgrade process, the *List Value Location* setting is changed from the default of Last to First to match the behavior from earlier releases.
5. Stop the source version of PingAccess.
6. Start the target version of PingAccess.

Warning text	Steps to take
`Resource '`*`ResourceName`*`' contains an invalid path prefix and cannot be migrated to the target version. Manual intervention is required.`	This occurs when the 2.1 path prefix contains functionality supported via a Java regex, but not by the wild card support in 3.1. The user must manually migrate the regex to 1 or more path prefixes in 3.1. For example, consider the 2.1 prefix, `/(app1|app2)`. This can be translated to a single resource in 3.1.1 with path prefixes of `/app1` and `/app2`.
`Resource '`*`ResourceName`*`' requires a case-sensitive path. This conflicts with its containing Application, which requires a case-insensitive path. Manual intervention may be required.`	The upgrade utility identifies path prefixes in 2.1 that start with `/(?i)` as path prefixes that are case-insensitive, and sets the case-sensitivity flag on the application appropriately. However, if multiple resources in a new application use inconsistent case sensitivity settings, the utility cannot determine what the case sensitivity should be. 2.1 resources are case-sensitive by default.
`Resource '`*`ResourceName`*`' requires a case-insensitive path. This conflicts with its containing Application, which requires a case-sensitive path. Manual intervention may be required.`	This is the same as the previous setting, but with the requirement being for a case-insensitive path rather than a case-sensitive one.
`Resource '`*`ResourceName`*`' is disabled in the source version. Resources can no longer be individually disabled. Application '`*`ApplicationName`*`' has been disabled due to this constraint.`	In 2.1, individual resources can be disabled. In 3.1, only applications can be enabled or disabled. The upgrade utility takes the approach of disabling the application if any related resources are disabled. Check the final configuration and make sure this is the desired outcome. If it is not, the disabled resources need to be deleted, and the application needs to be enabled.
`Path prefix for Resource '`*`ResourceName`*`' contains a '.' character. This will be treated as a literal '.' in the target version.`	In a 2.1 setup, there might be resource names that accidentally contain a '.', assuming it is a literal '.' rather than part of a regex. For example, any file extension type resources will probably not be escaping the '.'. This message is intended to bring this change in semantics to the user's attention. This action item will not show up if the user has correctly escaped the '.' character with the '\.' sequence.

Warning text	Steps to take
`Resource '`*`ResourceName`*`' could not be migrated to the target version due to Application context root conflicts. Manual intervention is required.`	This message indicates that multiple resources that use the same virtual host, but a different web session or site must be mapped under the same context root in the same application to preserve semantics. For example, consider the following configuration: ▪ Resource A: ▪ Path Prefix: /hr ▪ Virtual Host: internal.example.com ▪ Web Session: W ▪ Site: Z ▪ Resource B: ▪ Path Prefix: /sales ▪ Virtual host: internal.example.com ▪ Web Session: W ▪ Site: Z ▪ Resource C: ▪ Path Prefix: /payroll ▪ Virtual Host: internal.example.com ▪ Web Session: V ▪ Site: Z This configuration triggers this error because these resources cannot be grouped in the same application, but they would need to be in order to preserve the semantics in the internal.example.com address space. This issue could be fixed by using rewrite rules to place Resource C or Resources A and B under a different namespace. For example, use `/intranet/sales` and `/intranet/hr` on the front-end and rewrite out the `/intranet` on the backend.
`Application '`*`ApplicationName`*`' contains OAuth rules, but authenticates users with a web session. Unexpected results may occur.`	2.1 allows OAuth rules to be attached resources that use a web session. While this configuration is likely invalid in the first place, it would be possible to include both a PA cookie and OAuth token in requests and PA would apply policy to the requests as configured. In 3.1, however, an API application and web application are mutually exclusive so the semantics of this particular configuration cannot be preserved.

Warning text	Steps to take
`The resource order for Virtual Host '`*`VirtualHostName`*`' has changed in the target version.`	The upgrade utility checks that the resource order is consistent before and after the upgrade. This message indicates that the resource order from 2.1 does not match 3.1. This is likely due to how context roots in applications are ordered in 3.1. For 3.1, applications are ordered based on their context root, where the longest context root is checked first during resource matching. One way to address this is to review and potentially change the application context root values associated with the virtual host to avoid URL overlaps between applications.
`Application '`*`ApplicationName`*`' is no longer associated with an Identity Mapping. A Web Session or an Authorization Server is required to use Identity Mappings.`	Indicates a misconfiguration in the source version. Check whether you intended to use an Identity Mapping for the application and associate an appropriate Web Session or Authorization Server if necessary.
`OAuth Rule with id '`*`RuleId`*`' is no longer associated with Application '`*`ApplicationName`*`' because Application '`*`ApplicationName`*`' is not an OAuth Application. Manual intervention may be required.`	Indicates a misconfiguration in the source version. Check whether the OAuth Rule is necessary to implement the desired Access Control policy.
`OAuth RuleSet with id '`*`RuleSetId`*`' is no longer associated with Application '`*`ApplicationName`*`' because Application '`*`ApplicationName`*`' is not an OAuth Application. Manual intervention may be required.`	Indicates a misconfiguration in the source version. Check whether the OAuth RuleSet is necessary to implement the desired Access Control policy.
`Resource 'ResourceName' from Application with id '`*`ApplicationId`*`' was not migrated because the Application is a Web Application while the Resource has OAuth Rules. Manual intervention may be required.`	Indicates a Resource associated with the Application is associated with OAuth Rules. This is likely a misconfiguration, and it is necessary to evaluate whether this was intended or not.
`Upgrade created 'Availability Profile for Site '`*`SiteName`*`''. A more descriptive name may be required.`	Indicates that an Availability Profile was created for the Site during the upgrade. You may want to give the Availability Profile a more descriptive name.
`Application '`*`ApplicationName`*`' and associated Resources were not migrated. The context root of /pa is reserved. Manual intervention may be required.`	The */pa* context root was allowed as a valid context root in PingAccess 3.0 and is no longer allowed.
`Resource '`*`ResourceName`*`' from Application with id '`*`ApplicationId`*`' was not migrated because the /pa prefix is reserved when the Application context root is /. Manual intervention may be required.`	The */pa* path prefix was allowed as a valid path prefix in PingAccess 3.0 and is no longer allowed.

Warning text	Steps to take
`The OAuth Groovy Script Rule no longer controls the realm in the response sent for an unauthorized OAuth request.`	With PingAccess 3.2, Realms have been moved to the Application. The *Realm* can still be set using the PingAccess Admin API interface. With the change in context for how realms are applied, it is necessary to check existing OAuth Groovy Rules to ensure that they behave as expected. This message is shown if any OAuth Groovy Rules exist in the migrated configuration.
`The property '`*PropertyName*`' was set to a blank value to maintain compatibility. However, it is recommended that this be set to '`*PropertyName*`=`*PropertyValue*	New Security Headers properties values are not set during an upgrade in order to preserve the behavior from the source release in the upgrade. If there is no reason not to in your environment, update `run.properties` with the recommended setting.
`As a security enhancement, the default value of '`*CipherList*`' has changed with this version of PingAccess. Your existing ciphers remain unchanged. However, it is recommended to use the default value: '`*PropertyName*`=`*CipherList*`'.`	This message applies to the `admin.ssl.ciphers`, `engine.ssl.ciphers`, and `agent.ssl.ciphers` lists. This message is displayed if the upgrade source version cipher lists are changed from the defaults. We recommend updating the configuration with the new default value if possible.
`The property '`*PropertyName*`' was set to a blank value to maintain compatibility. However, it is recommended that this be set to '`*PropertyName*`=`*CipherList*	This message applies to the `site.ssl.protocols`, `site.ssl.ciphers`, `pf.ssl.protocols`, and `pf.ssl.ciphers` settings. The upgrade utility sets these values as empty values in order to maintain backwards compatibility, but the recommended value should be used if possible.
`The host for VirtualHost `*VirtualHost*`:`*Port*` already has a KeyPair associated with it. The KeyPair previously associated with this VirtualHost was removed. Only one KeyPair can be associated with a given host.`	If a Virtual Host has more than one key pair associated with it, only one key pair will be associated with it after the upgrade completes. This message is displayed to indicate which key pair was used.
`Application with name '`*ApplicationName*`' not migrated as the context root '`*Path*`' was a reserved path.`	If an application's context root is a reserved PingAccess path, the application will not be migrated. The indicated application will need to be created with a context root that does not conflict with the reserved path.
`Resource with name '`*ResourceName*`' not migrated as the path '`*Path*`' was a reserved path.`	If a Resource path is a reserved PingAccess path, the application will not be migrated. The indicated application will need to be created with a context root that does not conflict with the reserved path.
`The CIDR Rule with name '`*RuleName*`' is associated with an Agent Application named '`*ApplicationName*`' and overrides the IP source configuration. A new Agent rule should be created that does not override the IP source.`	With changes in IP source header handling, additional options are available to override the headers used to identify the source address. When an agent is involved, the changes in IP source handling may cause the specified rule to not behave as expected.

Warning text	Steps to take
`Require HTTPS option on Application '`*`ApplicationName`*`' was set to `*`Setting`*` as Virtual Host had port `*`Port`*`. Please verify this setting is correct.`	The upgrade utility attempts to set the `Require HTTPS` option based on the virtual host associated with an application during an upgrade. This message is an advisory to just verify that the setting was properly detected.
`VirtualHost '`*`VirtualHost`*`' was not migrated. An existing VirtualHost existed with the same logical name '`*`VirtualHost`*`'.`	Virtual host names are now case-insensitive. During the upgrade, after making the names case-insensitive, a duplicate virtual host was identified. It will be necessary to either recreate the virtual host with a new name, or to modify the configuration so the proper virtual host is migrated to the upgraded system.
`Renamed Virtual Host's Hostname from '`*`VirtualHost`*`' to '`*`NewVirtualHost`*`' due to virtual host spec compliance issue`	If a Virtual Host name contains an underscore (_) character, that does not conform to host naming requirements. In this instance, the underscore will be renamed to the string *a-z*. For example, if a Virtual Host named *my_virtual_host* is migrated, the new name will be *mya-zvirtuala-zhost*.
`Removed Http Request Rule with name '`*`RuleName`*`', this Rule must be converted to a groovy script rule. Manual intervention may be required.`	When an HTTP request rule is migrated from an earlier release of PingAccess, rules that specify a source of *Body* are not migrated. A Groovy script rule can be used to perform a similar match, but the details of such a Groovy script require administrator intervention. A simple Groovy script rule that would perform a similar function might be: `requestBodyContains('`*`value`*`')` We advise, however, that a script be constructed that performs additional validation in order to ensure the rule passes only when desired; a generic match like this could lead to unexpected results depending on what content might be in the request body.

Warning text	Steps to take
`The property '`*`PropertyName`*`' uses a customized value. "Your original value has not been modified. You may encounter startup or connection problems if this value is not supported by the JVM.`	When migrating SSL settings between versions of PingAccess that use different JVM or JDK versions, custom settings might not be compatible. If the protocols or ciphers used are not compatible with the target JVM or JDK, this message indicates which settings need to be manually updated. The *PropertyName* value can be any of the following values: ▪ `site.ssl.protocols` ▪ `site.ssl.ciphers` ▪ `pf.ssl.protocols` ▪ `pf.ssl.ciphers` ▪ `admin.ssl.protocols` ▪ `admin.ssl.ciphers` ▪ `engine.ssl.protocols` ▪ `engine.ssl.ciphers` ▪ `agent.ssl.protocols` ▪ `agent.ssl.ciphers`

`Rule with ID `*`RuleId`*` and name '`*`RuleName`*`' was not migrated as matcher was invalid for the Groovy rule type.`
 	
`Invalid rules were removed from RuleSet '`*`RuleSetName`*`' which resulted in an empty set.`
 	
The RuleSet was removed. Please check your policy configuration.
 	
`Invalid rules were removed from RuleSet '`*`RuleSetName`*`'. Please check your policy configuration.`
 	
`Invalid Rules were removed from Application '`*`ApplicationName`*`'. Please check your policy configuration.`
 	
`Invalid RuleSets were removed from Application '`*`ApplicationName`*`'. Please check your policy configuration.`
 	
`Invalid Rules were removed from Resource '`*`resource name`*`' on Application '`*`ApplicationName`*`'. Please check your policy configuration.`
 	
`Invalid RuleSets were removed from Resource '`*`resource name`*`' on Application '`*`ApplicationName`*`'. Please check your policy configuration.`
	These messages may be displayed if the source PingAccess installation has misconfigured Groovy Rules.
	
	This indicates that you are not permitted to add an OAuth rule to an Application of type Web, by editing an existing Rule Set.
	
	Groovy or OAuth Groovy rules will not be migrated for the following reasons:
	▪ The OAuth Groovy rule was applied to a Web application.
	▪ The Groovy or OAuth Groovy uses a matcher that is not appropriate for the application type.
	
	Check the policy configuration.

Warning text	Steps to take

`Rule with name '`*`RuleName`*`' has been removed from RuleSet with name '`*`RuleSetName`*`'. Multiple Rate Limiting Rules with the same Policy Granularity cannot be included in a RuleSet."`
 	
`Rule with name '`*`RuleName`*`' has been removed from RuleSet with name '`*`RuleSetName`*`'. Multiple Cross-Origin Request Rules cannot be included in a RuleSet."`
	The upgrade utility supports migrating a RuleSet containing multiple CORS or Rate Limiting rules with the same Policy Granularity. The upgrade utility will generate new action items, indicating that rules were removed from a RuleSet.
	
	These messages indicate that if both rules exist, there is a restriction to a single Rate Limiting or CORS rule. Please check to confirm that you have applied the correct rule to the policy.

`One or more notifications were issued while migrating from version `*`SOURCE`*` to version `*`TARGET`*
 	
`Setting clusterconfig.enabled to false`
 	
`The new configuration query port feature has been disabled for backward compatibility. Please refer to the PingAccess clustering documentation before enabling this feature.`
	The new cluster config query port is enabled by default for new PingAccess 4.0 installations when running in CLUSTERED_CONSOLE or CLUSTERED_CONSOLE_REPLICA mode.
	
	During the upgrade process to version 4.0, the new cluster config query port is disabled. Messages are written to `upgrade.log` and `audit.log` to indicate this cluster configuration change was made.
	
	Please refer to the PingAccess clustering documentation before enabling this feature.

`One or more notifications were issued while migrating from version `*`SOURCE`*` to version `*`TARGET`*
 	
`For backward compatibility, when connecting to a protected, TLS SNI-enabled Site, PingAccess will set the SNI server_name to the configured target host and not the HTTP request Host header value. Please refer to PingAccess' upgrade documentation for more information.`
	During upgrades to release 4.0 and higher, the Upgrade Utility sets the value of `pa.site.tls.sni.legacyMode` to `true` to maintain compatibility with existing configurations. This property is controlled in the `run.properties` file and is not enabled on new installs.

`Localization property '`*`{property name}`*`' was added to pa-messages.properties. Any customized localization files should be updated.`	This message will appear if new language properties are added between the source and target PA versions and you have added additional language files or modified the en or en_US files. Update any customized files as required.
`Localization property '`*`{property name}`*`' in pa-messages.properties was modified. Any customized localization files should be updated.`	This message will appear if the language properties have changed between the source and target PA versions and you have added additional language files or modified the en or en_US files. Update any customized files as required.
`Localization property '`*`{property name}`*`' was removed from pa-messages.properties. This property can be removed from any customized localization files.`	This message will appear if the language properties have been removed between the source and target PA versions and you have added additional language files or modified the en or en_US files. Update any customized files as required.

Warning text	Steps to take
`WebSessionManagement contained an invalid cookie name. Replaced '{old cookie name}' with '{new cookie name}'. Please validate your configuration.`	This message will appear if the WebSessionManagement has an invalid cookie name. Invalid characters are replaced with an underscore. Update any references as required.
`Legacy authentication requirements policy evaluation has been enabled to maintain backward compatibility with earlier versions of PingAccess. To disable this setting, remove the pa.policy.eval.acr.v42 property from run.properties.`	This message will appear on upgrade to release 4.3 or higher if you have one or more authentication requirements rules. You can make adjustments to configured rules so you can remove this property or you can maintain the property to leave existing rules unaffected.
`Property pa.audit.log.applicationResourceIdsAsIntegers was set to true in run.properties to maintain existing behavior. In order to log the ID of Global Unprotected Resources, this property should be removed or should be set to false (default). However, a value of false (default) will result in resourceId and applicationId audit logging fields being logged as strings, not integers, which may require audit logging database schema changes if these values are currently being used.`	This message will appear on upgrade to release 5.1 or higher to support the existing logging behavior of application resource IDs as integers. The default behavior of release 5.1 and higher is to log these IDs as strings. You can choose to log application resource IDs as strings after the upgrade by removing, or setting to false, the applicable property in run.properties. This change may require a modification to the audit logging database schema.
`Invalid resource method 'Method' was removed from Resource 'ResourceName' on Application 'ApplicationName'.`	This message will appear on upgrade to release 5.3 or later if the source version has an application resource that contains a method with whitespace. The resource is preserved by the upgrade, but the method is removed.
`Invalid Resource {name} on Application {name} was removed because it did not have any valid methods.`	This message will appear on upgrade to release 5.3 or later if all of the methods associated with a resource were removed with an `Invalid resource method` error. The resource is not migrated by the upgrade.

Restore a PingAccess configuration backup

If an upgrade fails, you can restore your PingAccess configuration using an automatically generated backup.

About this task

PingAccess automatically creates a backup zip file each time an Administrative user authenticates to the administrative console. These backups are stored in *PA_HOME*`/data/archive`, with a maximum number of backups configurable using the `pa.backup.filesToKeep` configuration parameter in `run.properties`.

ⓘ **CAUTION:** This operation will replace your current configuration settings.

Steps

1. Stop PingAccess.
2. Unzip the backup file to *`PA_HOME`* .
3. Restart PingAccess.

Results
Your PingAccess configuration will now be reverted to the state in the backup archive that was restored.

Upgrade utility configuration file reference

This document provides a reference to configurable parameters used by the upgrade utility. These parameters are configured in the `run.properties` file located at `<UU_HOME>/conf/`.

pa.upgrade.source.ssl.ciphers

Defines the type of cryptographic ciphers available for use with the source PingAccess.

pa.upgrade.source.ssl.protocols

Defines the protocols available for use with the source PingAccess.

pa.upgrade.target.ssl.ciphers

Defines the type of cryptographic ciphers available for use with the target PingAccess. If not specified, the JVM default values are used.

pa.upgrade.target.ssl.protocols

Defines the protocols available for use with the target PingAccess. If not specified, the JVM default values are used.

pa.upgrade.http.client.connection.timeout.ms

Defines, in milliseconds, the amount of time to wait before timing out the connection to the HTTP client. The default value is 3600000.

pa.upgrade.http.client.socket.timeout.ms

Defines, in milliseconds, the HTTP client socket timeout. The default value is 3600000.

PingAccess: Zero Downtime Upgrade

Introduction

This document describes the steps required to perform a zero downtime upgrade of a PingAccess cluster to version 5.0 or higher. A zero downtime upgrade allows you to upgrade your environment with no impact to resource availability or existing user sessions.

Though this procedure is applicable to any PingAccess cluster upgrade to version 5.0 or higher, there are minor variations depending on your PingAccess source version. Those variations are clearly described where applicable.

Note that there are some steps, particularly those related to working with a load balancer, that are dependent on your environment. It is expected that you are familiar with the tasks required by these steps and, accordingly, this document does not attempt to offer detailed instruction on performing these tasks.

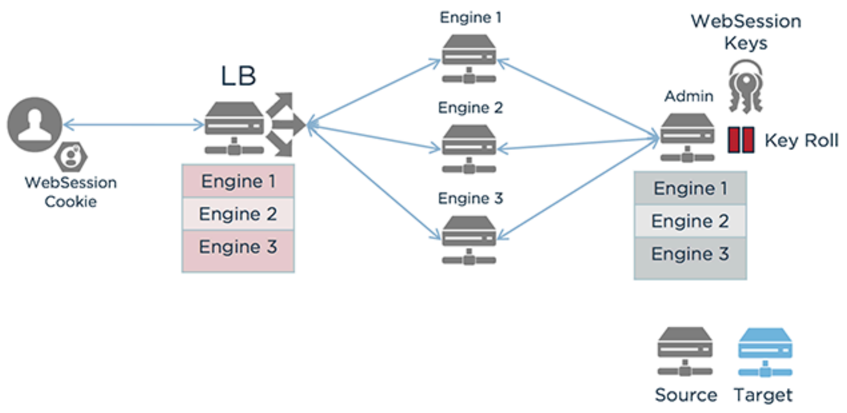
Important: In order to achieve a successful upgrade, perform the steps in this document in the order that they are presented. Deviation from these steps may result in a failed upgrade and/or system downtime.

Before you begin, review the *Upgrade considerations* on page 13.

To begin the upgrade process, *Disable key rolling* to prevent active sessions from being invalidated.

Step 1: Disable key rolling

In this step, you will disable key rolling to prevent active sessions from being invalidated during the upgrade process. This is a temporary modification that you will address when the upgrade is complete. Note that there are different procedures at this stage depending on your source version.



Source version of PingAccess 5.2 or later

If the source is PingAccess 5.2 or higher, you can disable key rolling.

1. Navigate to **Settings# Access# Identity Mappings**.
2. In the **Auth Token Management** section, deselect **Key Roll Enabled**.
3. Click **Save**.
4. Navigate to **Settings# Access# Web Sessions**.
5. In the **Web Session Management** section, deselect **Key Roll Enabled**.
6. Click **Save**.
7. Navigate to **Settings# System# Token Validation**.
8. In the **OAuth Key Management** section, deselect **Key Roll Enabled**.
9. Click **Save**.

Source version of PingAccess 5.0 or 5.1

If the source is PingAccess 5.0 or 5.1, you can disable key rolling.

1. Navigate to **Settings# Access# Identity Mappings**.
2. In the **Auth Token Management** section, deselect **Key Roll Enabled**.
3. Click **Save**.
4. Navigate to **Settings# Access# Web Sessions**.
5. In the **Web Session Management** section, deselect **Key Roll Enabled**.
6. Click **Save**.

Source is an earlier version of PingAccess

If the source is a version of PingAccess earlier than 5.0, you can set the key rolling interval to a value that allows enough time for the upgrade to be completed successfully.

1. Navigate to **Settings# Access# Identity Mappings**.
2. In the **Auth Token Management** section, specify a **Key Roll Interval** of 240 (10 days).
3. Click **Save**.
4. Navigate to **Settings# Access# Web Sessions**.

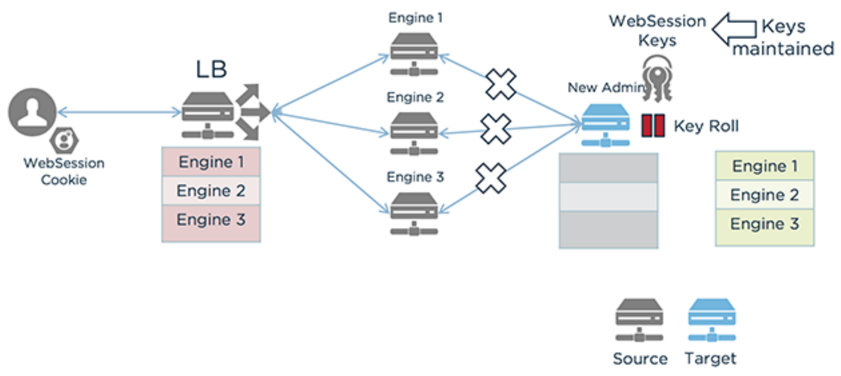
5. In the **Web Session Management** section, specify a **Key Roll interval** of 240 (10 days).
6. Click **Save**.

Next, you will *upgrade the Admin node*.

Step 2: Upgrade the Admin node

In this step, you will upgrade the PingAccess administration node using the PingAccess Upgrade Utility. As part of this step, you will use the `-r` switch to disable configuration replication on the target version.

For more information on upgrading PingAccess, see *Upgrade PingAccess*.



Prerequisites

Prior to beginning the upgrade process, make sure you have:

- Ensured PingAccess is running
- Downloaded and unpacked the PingAccess *Upgrade Utility*
- Downloaded the PingAccess *distribution ZIP file*
- The PingAccess license
- Administrator credentials
- Basic Authentication enabled

ⓘ **Warning:** Failure to enable basic authentication can result in losing access to the Administration console.

Upgrade the PingAccess administrative node

1. To run the Upgrade Utility, use a command line to change to the Upgrade Utility's `/bin` directory. For example:

```
cd /pa-upgrade-5.0.1/bin
```

2. Run the Upgrade Utility:
   - Linux: `./run.sh -r <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`

     For example:

     ```
     ./run.sh -r ../pingaccess-4.3.0 ../pingaccess-5.0.1 ../
     pingaccess-5.0.1.zip ../pingaccess.lic
     ```
   - Windows: `run.bat -r <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`

     For example:

     ```
     run.bat -r ../pingaccess-4.3.0 ../pingaccess-5.0.1 ../
     pingaccess-5.0.1.zip ../pingaccess.lic
     ```

   Important: The `-r` switch will disable configuration replication on the Admin node. You will re-enable configuration replication after the Admin and all engine nodes have been upgraded.
3. Stop the existing PingAccess admin instance.
4. Start the new PingAccess admin instance.
5. Upgrade the Admin Replica node using the same steps, but omitting the `-r` switch.

Important: If PingAccess is running as a service:

- In Linux, update **PA_HOME** in `/etc/systemd/system/pingaccess.service` to point to the new installation.
- In Windows, remove the existing PingAccess service (`<OLD_PA_HOME>\sbin\Windows\uninstall-service.bat`) and add the new service (`<NEW_PA_HOME>\sbin\Windows\install-service.bat`).

After you have upgraded the Admin and Replica Admin nodes, you can begin *upgrading the engines*.

Step 3: Upgrade engines

This phase of the zero downtime upgrade focuses on upgrading each engine in the cluster. To maintain resource availability, you perform this set of steps on **one engine at a time** until all engines are successfully upgraded.

Important: It is imperative that you perform these steps on one engine at a time to maintain availability. Engines are identified by the engine name. Ensure that the engine that you remove from the load balancer aligns with the engine definition you import.

This phase requires that the following steps take place for each engine in the cluster, **one at a time**:

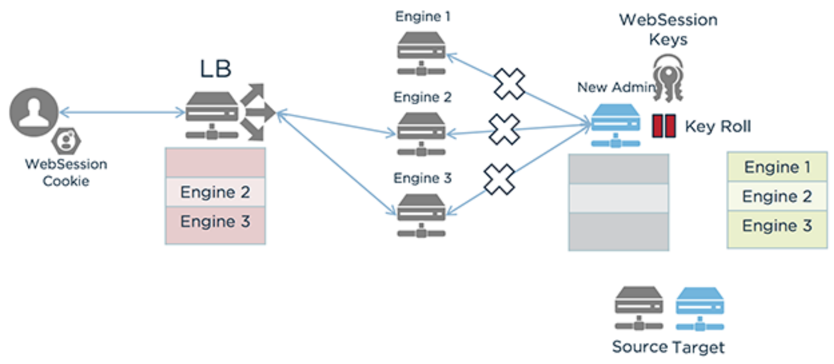
- *Remove the engine from the load balancer*
- *Upgrade the engine*
- *Resume configuration replication* on page 72
- *Add the engine to the load balancer*

Important: Do not begin the upgrade of an additional engine until the active engine upgrade is completed and the engine is reporting to the PingAccess administrative node.

Remove the engine from the load balancer configuration

This step requires you to remove the engine from the load balancer configuration. Since this step is dependent on your environment, no specific instruction will be provided. It is assumed that you are familiar with the steps required to temporarily remove the engine from your load balancer configuration.

ⓘ **Important:** To maintain resource availability, you should remove only the engine you are upgrading. After the upgrade is complete, you will add the engine back to the load balancer configuration. Only after you confirm that the engine has been successfully added to the load balancer and is reporting properly to PingAccess should you begin the upgrade process on additional engines.



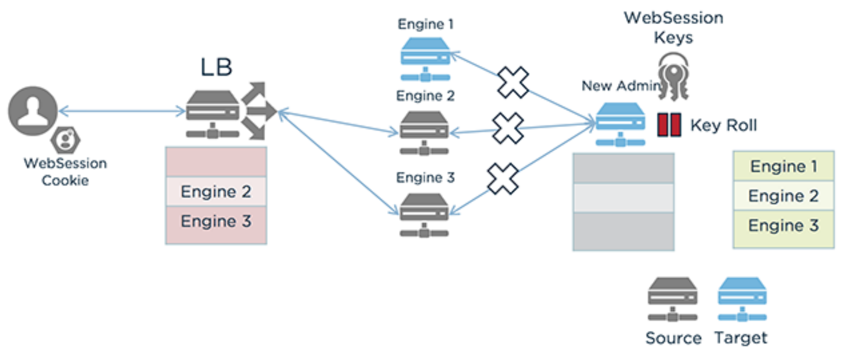
Remove the engine from the load balancer configuration

1. Identify and note the engine you want to upgrade. Ensure you have the engine definition for this engine available.
2. Locate the entry for this engine in your load balancer configuration.
3. Depending on your environment, you may be removing this entry or you may be commenting the entry to remove it. Ensure that you make note of any entries you remove or modify so that you can reverse this operation later in *Add the engine to the load balancer configuration* on page 73.
4. Remove the engine entry from the load balancer.
5. Restart the load balancer.

Upgrade the engine

In this step, you will use the PingAccess Upgrade Utility to upgrade the engine.

For more information on upgrading PingAccess, see *Upgrade PingAccess*.



Prerequisites

Prior to beginning the upgrade process, make sure you have:

- Ensured the PingAccess engine is running

- Downloaded and unpacked the PingAccess *Upgrade Utility*
- Downloaded the PingAccess *distribution ZIP file*
- The PingAccess license

Upgrade the PingAccess engine

1. To run the Upgrade Utility, use a command line to change to the Upgrade Utility's `/bin` directory. For example:

   ```
   cd /pa-upgrade-5.0.1/bin
   ```
2. Run the Upgrade Utility:
   - Linux: `./run.sh <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`

     For example:

     ```
     /run.sh ../pingaccess-4.3.0 ../pingaccess-5.0.1 ../
     pingaccess-5.0.1.zip ../pingaccess.lic
     ```
   - Windows: `run.bat <sourcePingAccessRootDir> <outputDir> <pingaccessZip> <newPingAccessLicense>`

     For example:

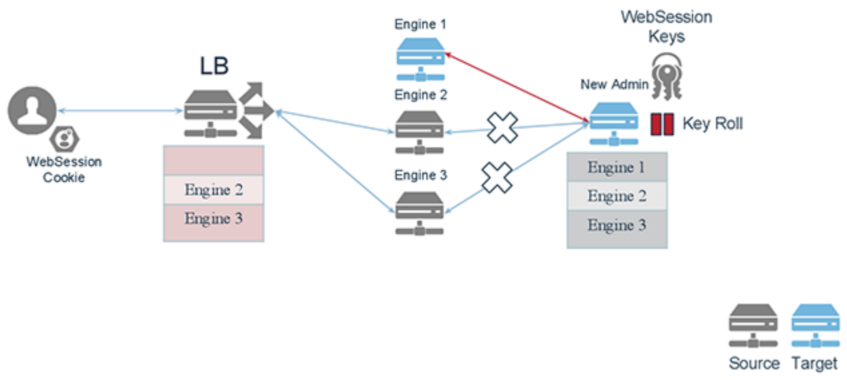
     ```
     run.bat ../pingaccess-4.3.0 ../pingaccess-5.0.1 ../
     pingaccess-5.0.1.zip ../pingaccess.lic
     ```
3. Stop the existing PingAccess instance. Do not start the new instance.

Important: If PingAccess is running as a service:

- In Linux, update **PA_HOME** in `/etc/systemd/system/pingaccess.service` to point to the new installation.
- In Windows, remove the existing PingAccess service (`<OLD_PA_HOME>\sbin\Windows\uninstall-service.bat`) and add the new service (`<NEW_PA_HOME>\sbin\Windows\install-service.bat`).

Resume configuration replication

In this step, you will resume configuration replication that was disabled by the Upgrade Utility. You will perform this step for the Admin replica and all engines in the cluster. You will use the PingAccess Admin API to GET and PUT the relevant configuration data for each of these items.



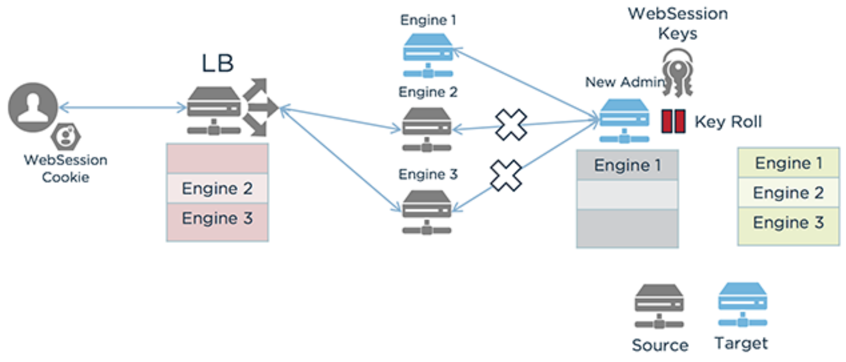
To resume configuration replication:

ⓘ **Note:** Perform the following steps for the Replica Admin node and for each engine in the cluster.

1. In a browser, navigate to `https://<PingAccessHost>:9000/pa-admin-api/v3/api-docs/`.
2. For engines, expand the **/engines** endpoint.
3. Click the **GET /engines** operation.
4. Click **Try it out!** and note the engine `id` for each engine.
5. Click the **GET /engines/{id}** operation.
6. Enter the `id` of the engine you want to update and click **Try it out!**
7. Copy the **Response Body**.
8. Click the **PUT /engines/{id}** operation and enter the `id` of the engine you want to update.
9. Paste the **Response Body** you copied and change "`configReplicationEnabled`" to `true`.
10. Click **Try it out!**

    If the operation is successful, you will receive a **Response Code** of **200**.
11. Repeat the previous steps for each engine.
12. In the PingAccess administration console, navigate to **Settings# System# Clustering**.
13. Ensure the engines are displayed and reporting. A healthy engine shows a green status indicator.

ⓘ **Note:** There may be a delay in bringing the engine to a running status. If the engine does not immediately show as reporting, refresh the page until the engine status indicator is green (running).



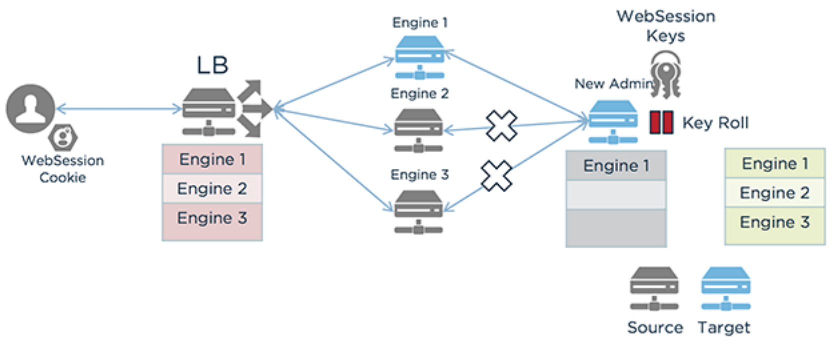
14. Expand the **/adminConfig/replicaAdmins** endpoint.
15. Click the **GET /adminConfig/replicaAdmins** operation.
16. Click **Try it out!** and note the `id` for the replica admin.
17. Click the **GET /adminConfig/replicaAdmins/{id}** operation.
18. Enter the `id` of the replica admin you want to update and click **Try it out!**
19. Copy the **Response Body**.
20. Click the **PUT /adminConfig/replicaAdmins/{id}** operation and enter the `id` of the replica admin you want to update.
21. Paste the **Response Body** you copied and change "`configReplicationEnabled`" to `true`.
22. Click **Try it out!**

    If the operation is successful, you will receive a **Response Code** of **200**.
23. In the PingAccess administration console, navigate to **Settings# System# Clustering**.
24. Ensure the **Replica Administrative Node** displayed and reporting on the **Administrative Nodes** screen. A healthy node shows a green status indicator.

Add the engine to the load balancer configuration

This step requires you to add the engine back to the load balancer configuration. Since this step is dependent on your environment, no specific instruction will be provided. It is assumed that you are familiar with the steps required to add the engine back to the load balancer configuration.

After you confirm that the engine has been successfully added to the load balancer and is reporting properly to PingAccess, you can begin the upgrade process on additional engines.



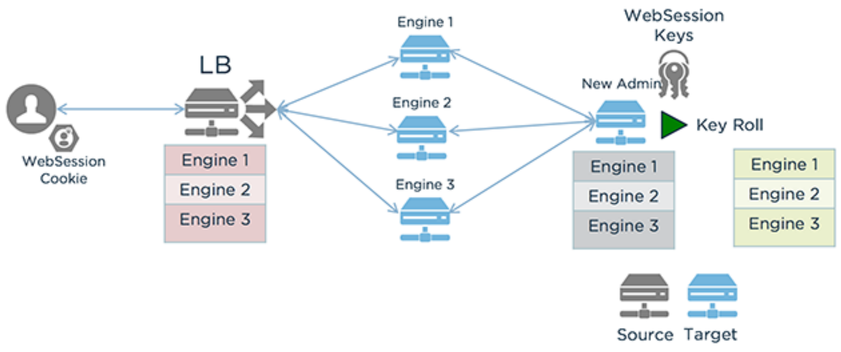
Add the engine to the load balancer configuration

1. To add the engine to the load balancer configuration, simply reverse the steps you took in *Remove the engine from the load balancer configuration* on page 71 to remove the engine.
2. Restart the load balancer.

Repeat the *Step 3: Upgrade engines* on page 70 process until each engine has been upgraded.
When all engines have been upgraded, added to the load balancer configuration, and are reporting to PingAccess, you can move on to the final step, *Enable key rolling*, to complete the zero downtime upgrade process.

Step 4: Enable key rolling

In this step, you will resume key rolling. Note that there are different procedures at this stage depending on your source version.



Source version of PingAccess 5.2 or later

1. Navigate to **Settings# Access# Identity Mappings**.
2. In the **Auth Token Management** section, select **Key Roll Enabled**.
3. Click **Save**.
4. Navigate to **Settings# Access# Web Sessions**.
5. In the **Web Session Management** section, select **Key Roll Enabled**.
6. Click **Save**.
7. Navigate to **Settings# System# Token Validation**.
8. In the **OAuth Key Management** section, select **Key Roll Enabled**.
9. Click **Save**.

Source version of PingAccess 5.0 or 5.1

1. Navigate to **Settings**# **Access**# **Identity Mappings**.
2. In the **Auth Token Management** section, select **Key Roll Enabled**.
3. Click **Save**.
4. Navigate to **Settings**# **Access**# **Web Sessions**.
5. In the **Web Session Management** section, select **Key Roll Enabled**.
6. Click **Save**.

Source is an earlier version of PingAccess

1. Navigate to **Settings**# **Access**# **Identity Mappings**.
2. In the **Auth Token Management** section, specify a **Key Roll Interval** of your choosing. The default value is 24 (1 day).
3. Click **Save**.
4. Navigate to **Settings**# **Access**# **Web Sessions**.
5. In the **Web Session Management** section, specify a **Key Roll interval** of your choosing. The default value is 24 (1 day).
6. Click **Save**.

Recovering from a failed upgrade

The Zero Downtime Upgrade process creates a set of new folders for the upgraded installation. The pre-upgrade source installation is not affected.

To recover your PingAccess cluster in the event of a failure, you would utilize the former installation by:

1. Stopping any upgraded PingAccess instances.
2. Start the original PingAccess instance on the Admin node.
3. Import the engine definitions back into the original PingAccess instance.
4. Start the original PingAccess instances on the Engine nodes.
5. Ensure all engines are added to the load balancer configuration.

Reference Guides

API Endpoints

PingAccess endpoints

These endpoints enable external applications to communicate with the PingAccess server and provide complete administrative capabilities of the product.

Heartbeat endpoint

Enables administrators to verify that the server is running.

OpenID Connect endpoints

Enable PingFederate or other token providers to interface with PingAccess using the OpenID Connect (OIDC) protocol.

Authentication Token Management endpoint

Enables protected applications to validate authentication tokens issued by a PingAccess identity mapping.

OAuth endpoint

Enables an OAuth Authorization Server to interface with PingAccess as an OAuth Resource Server.

Administrative API endpoints

Enable users to use PingAccess administrative functions.These are REST APIs that include documentation and testing tools.

> **Important:** Some endpoint examples in this document include the `/pa` reserved path. This document assumes the default application reserved path has not been modified. You can modify the reserved path using the *PingAccess Admin API*. If the reserved path has been modified, update endpoint and other applicable application URLs appropriately.

Heartbeat endpoint

This page describes the endpoint used to verify that the PingAccess server is running and, depending on security settings, view details about the configuration.

You can make this call to any active PingAccess listener and on any node in a PingAccess cluster. For example, with default port configurations, a Clustered Console Replica will respond to this endpoint on port 9000, and a Clustered Engine will respond to it on port 3000.

/pa/heartbeat.ping

This endpoint returns a short or detailed status for the target PingAccess server, based on the value of the `enable.detailed.heartbeat.response` parameter in run.properties. Load balancers can use this endpoint to determine the status of PingAccess.

> **Info:** Begin the URL with the server name and the PingAccess runtime port number. For example: `https://`*`hostname`*`:3000/pa/heartbeat.ping`.

If an error is returned, this indicates that the PingAccess instance associated with the endpoint is down.

If `enable.detailed.heartbeat.response` is set to `false` (the default value) and the PingAccess instance is running, the endpoint returns an HTTP `200` status and the text `OK`.

If `enable.detailed.heartbeat.response` is set to `true` and the PingAccess instance is running, a configurable status with additional details is returned. The response output format is an Apache Velocity template defined in `PA_HOME/conf/template/heartbeat.page.json`. You can modify this template to suit your needs. The following values are available:

Value	Description
$monitor.getTotalJvmMemory('bytes'|'KB'|'MB'|'GB')	Returns the total memory in the JVM. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.
$monitor.getUsedJvmMemory('bytes'|'KB'|'MB'|'GB')	and the PingAccess instance is running, a configurable status with additional details isReturns the used memory in the JVM. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.
$monitor.getFreeJvmMemory('bytes'|'KB'|'MB'|'GB')	Returns the free memory in the JVM. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.
$monitor.getTotalPhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')	Returns the total system memory. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.

Value	Description
$monitor.getTotalUsedPhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')	Returns the used system memory. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.
$monitor.getTotalFreePhysicalSystemMemory('bytes'|'KB'|'MB'|'GB')	Returns the free system memory. Specify 'bytes', 'KB', "MB', or 'GB' to specify the units. 'bytes' is the default if not specified.
$monitor.getHostname()	Returns the hostname for the system running PingAccess.
$monitor.getNumberOfCpus()	Returns the number of CPU cores in the system.
$monitor.getCpuLoad('###.##')	Returns the current CPU utilization. The parameter contains an optional format value. If the format is specified, the value returned is returned as a percentage value from 0%-100%, formatted using the *Java DecimalFormat* specification. If no format value is specified, then the value returned is a real number from 0 to 1 which represents the CPU utilization percentage. For example, a format value of "###.##" will return a value similar to "56.12", but no specified format would result in the value being returned as "0.5612".
$monitor.getOpenClientConnections()	Returns the current number of clients connected to PingAccess.
$monitor.getNumberOfVirtualHosts()	Returns the current number of configured virtual hosts in PingAccess.
$monitor.getNumberOfApplications()	Returns the current number of configured applications in PingAccess.
$monitor.getNumberOfSites()	Returns the current number of configured sites in the PingAccess configuration database. In a clustered environment, on the engine nodes, this number will reflect the number of sites associated with applications rather than the number of configured sites that show on the admin node. For more information, see **Server Clustering** documentation. This value is not included in the default template, but can be added by the system administrator if desired.
$monitor.getLastRefreshTime('yyyy/MM/dd HH:mm:ss')	Returns the time the PingAccess configuration was last refreshed. The parameter specifies the date format to use; if no value is specified, the *ISO 8601* date format is used. If the parameter is specified, the format used comes from the *Joda DateTimeFormat* specification.

The default content type for the output is application/json. However, you can specify a content type header using the `$monitor.setContentType()` line in the template.

If you update the `enable.detailed.heartbeat.response` value, you must restart PingAccess to make the new value take effect.

OpenID Connect endpoints

This page describes the endpoints needed for PingFederate or another token provider to interface with PingAccess using the OpenID Connect (OIDC) protocol.

These endpoints are available on the `engine.http.port` and `agent.http.port` ports defined in `PA_HOME/conf/run.properties`.

/pa/oidc/logout

This endpoint clears the browser cookie containing the PA Token. This endpoint enables end users to trigger the removal of their own PA Cookie from the browser they are using. The user is redirected to the *Logged Out* page. You can modify the template for this page, located at `<PA_INSTALL>/conf/template/general.loggedout.page.template.html`.

ⓘ **Info:** This endpoint does not retain any server-side state to denote log off. Additionally, this endpoint clears the cookie only from the requested host/domain, and the cookie may still exist in requests bound for other hosts/domains.

ⓘ **Note:** If you want to log out across multiple domains, you can use the PingFederate `/idp/startSLO.ping` endpoint instead. See the *PingFederate documentation* for more information about this endpoint.

/pa/oidc/cb

This endpoint receives the ID Token from the token provider.

/pa/oidc/JWKS

This endpoint is used by the token provider's JWT Token Processor for signature verification. This endpoint must be used in conjunction with the configuration of a JWT token processor instance in the token provider. For more information on configuring a JWT in PingFederate, see the *PingFederate documentation*.

/pa/oidc/logout.png

This endpoint is used by the token provider to initiate a logout from PingAccess in conjunction with the single logout functionality, terminating the PA tokens across domains.

Authentication Token Management endpoint

This page describes the endpoint used for Authentication Token Management.

/pa/authtoken/JWKS

This endpoint is used by backend sites to validate the signature of a JWT. For more information on JWT, see the *OpenID Connect 1.0 Developers Guide*.

OAuth endpoint

This page describes the endpoint used by an OAuth authorization server to interface with PingAccess as an OAuth resource server.

/pa/oauth/JWKS

This endpoint is used by an OAuth authorization server to acquire PingAccess public keys for encryption of access tokens. The output uses the IETF JWK format for public keys.

Administrative API endpoints

This page describes the endpoints available for administering PingAccess.

PingAccess ships with interactive documentation for both developers and non-developers to explore the PingAccess API endpoints, view a reference of the metadata for each API, and experiment with API calls. PingAccess APIs are REST APIs that provide complete administrative capabilities of the product. They can be called from custom applications or from command line tools such as cURL.

These endpoints are only available on the `admin.port` defined in *`PA_HOME`*`/conf/run.properties` at path `/pa-admin-api/v3/api-docs/` (`https://<`*`PA_HOME`*`>:<`*`PORT`*`>/pa-admin-api/v3/api-docs/`).

Note: For enhanced API security, you must include `X-XSRF-Header: PingAccess` in all requests and use the `application/json` content type for PUT/POST requests.

Clustering Reference Guide

Clustering

PingAccess can be configured in a clustered environment to provide higher scalability and availability for critical services.

PingAccess clusters are made up of three types of nodes:

Administrative Node

Provides the administrator with a configuration interface.

Replica Administrative Node

Provides the administrator with the ability to recover a failed administrative node using a manual failover procedure.

Engine Node

Handles incoming client requests and evaluates policy decisions based on the configuration replicated from the administrative node.

Any number of engine nodes can be configured in a cluster, but only one administrative node and one replica administrative node can be configured in a cluster.

You should manage incoming traffic to the engine nodes using load balancers or other mechanisms. PingAccess clusters do not dynamically manage or load-balance request traffic to individual engine nodes.

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.

Configuration information is replicated to all of the engine nodes and the replica administrative node from the administrative node. State information sharing between engine nodes is not part of a default cluster configuration. However, some environments can benefit from Runtime State Clustering, which is an optional function that lets engine nodes replicate and share some state information with each other.

The license file on the administrative node is replicated to all of the engine nodes and the replica administrative node. The engine nodes do not require a license to function, but some default templates appear differently depending on the information in the license.

Node failure implications

The failure of a node within a PingAccess cluster can have short-term or long-term implications, depending on the node and your network state.

Node issues

Node issue	Result	Recommendation
Administrative node failure	The engine nodes function using stored configurations, but cannot update their configurations.	Fail over to the replica administrative node until the administrative node can be restarted.
Replica administrative node failure	The engine nodes and administrative node function normally. No failover is available in case of administrative node failure.	Restart the replica administrative node as soon as possible.
Administrative and replica node failure	The engine nodes function using stored configurations, but cannot update their configurations. No failover is available.	Restart the administrative node as soon as possible, or restart the replica administrative node and fail over.
Some engine nodes cannot reach the administrative node	Affected engine nodes function using stored configurations, if any, but cannot update their configurations. If the administrative node performs key rolling, the affected engine nodes cannot recognize the new PingAccess internal cookie.	Restore administrative node access as soon as possible.

Cluster properties

Use the `run.properties` and `bootstrap.properties` files to configure your environment.

In a cluster, you can configure each PingAccess node to serve as either an administrative node, a replica administrative node, or an engine node in the `run.properties` file. The `run.properties` file for the administrative node also contains server-specific configuration data.

At startup, a PingAccess engine node in a cluster checks its local configuration and then makes a call to the administrative node to check for changes. You can configure how often each engine node in a cluster checks the administrative node for changes in the engine `run.properties` file.

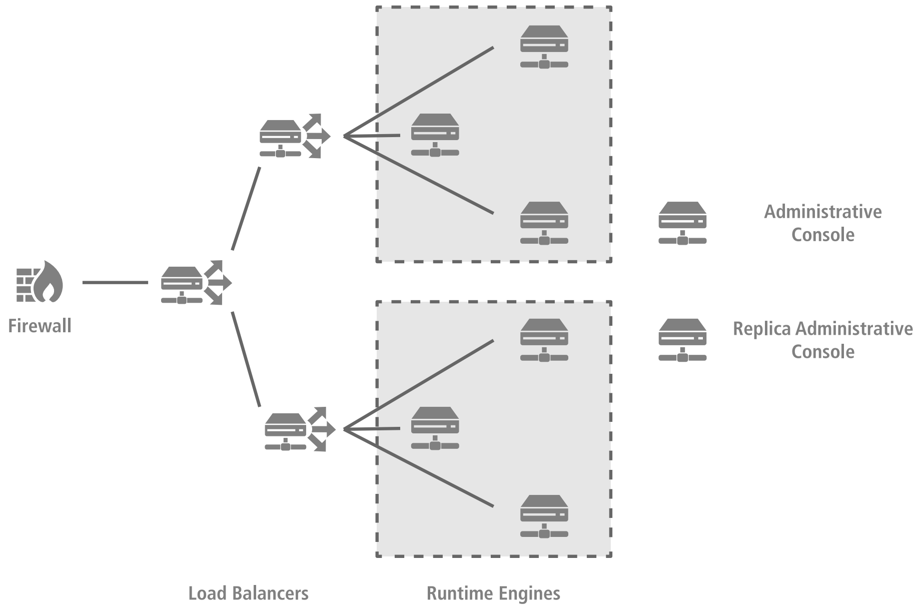
Configuration information is replicated to all engine nodes. By default, engines do not share runtime state. You can configure nodes for Runtime State Clustering using the `run.properties` file.

Information needed to bootstrap an engine node is stored in the `bootstrap.properties` file on each engine node.

bootstrap.properties

engine.admin.configuration. host	Defines the host where the administrative console is available. The default is `localhost`.
engine.admin.configuration. port	Defines the port where the administrative console is running. The default is `9000`.
engine.admin.configuration. userid	Defines the name of the engine.
engine.admin.configuration. keypair	Defines an elliptic curve key pair that is in the JSON Web Key (JWK) format.
engine.admin.configuration. bootstrap.truststore	Defines the truststore, in JWK format, that is used for communication with the administrative console.

ⓘ **Info:** The cache can be tuned using the EHCache Configuration Properties (`pa.ehcache.*`) listed in the *Configuration file reference guide*.



Cluster node status

Engine nodes and replica administrative nodes include a status indicator that indicates the health of the node and a **Last Updated** field that indicates the date and time of the last update. The status indicator can be green (good status), yellow (degraded status), or red (failed status).

The status is determined by using the value for `admin.polling.delay` as an interval to measure health:

Green (good status):

The node contacted the administrative node on the last pull request.

Yellow (degraded status):

The node contacted the administrative node between 2 and 10 intervals.

Red (failed status):

The node has either never contacted the administrative node, or it has been more than 10 intervals since the nodes communicated.

Using multiple network interface cards to route traffic

The routing of different types of traffic over specific interfaces is a network infrastructure exercise. However, PingAccess does support the routing of traffic over multiple network interfaces since, by default, PingAccess binds to all interfaces, as specified by a 0.0.0.0 address for the following parameters in `conf/run.properties`.

```
admin.bindAddress=0.0.0.0
clusterconfig.bindAddress=0.0.0.0
engine.http.bindAddress=0.0.0.0
agent.http.bindAddress=0.0.0.0
```

You can override this setting by specifying a single bind address.

Runtime state clustering

Runtime state clustering provides better scaling of large PingAccess deployments by allowing multiple engine nodes in the configuration to share certain information. A load balancer is placed in front of each group of nodes in order to distribute connections to the nodes.

Runtime state clustering serves three purposes:

- Providing fault-tolerance for mediated tokens if an engine node is taken offline.
- Reducing the number of STS transactions with PingFederate when the front-end load balancer does not provide a sticky session.
- Ensuring rate limits are enforced properly if the front-end load balancer does not provide a sticky session.

Runtime state clustering is not necessary in most environments. It can be beneficial in very large environments or environments using rate limiting rules or token mediation.

Configure a PingAccess cluster

This procedure installs and configures PingAccess on each cluster node, including the administrative node, an optional replica administrative node, and one or more engine nodes. The initial node becomes the administrative node, and is used to configure the rest of the cluster.

About this task

The configuration includes setting the `pa.operational.mode` property on each node. Do not modify this property until directed to do so.

Steps

1. Install PingAccess on each cluster node.

   Note: The sequence of the configuration flow is significant: host names first, then generate keys, and then configure the cluster.

2. Create a key pair for the PingAccess administrative console that uses the DNS name of the administrative node as the common name. If an administrative replica console is created, this key pair needs to either be configured with the administrative replica node defined in the Subject Alternative Names for the key pair, or the key pair needs to be configured as a wildcard certificate.

   Info: Using an IP address as the common name or in the subject alternative names is also acceptable, as long as those values are used in the administrative node fields on the **Administrative Nodes** configuration page.

3. Go to **Settings**# **System**# **Clustering**# **Administrative Nodes** and define the **Primary Administrative Node** as a *host*:*port* pair.

   The *host* must be a resolvable DNS name for the node or the node's IP address. The *port* is the TCP port PingAccess listens to for the administrative interface. The default port is `9090`.
4. If a replica administrative node will be used in the cluster, perform the following steps:
   a. Define a *host*:*port* pair for the replica administrative node.
   b. Click **Save & Download** to download the replica administrative node configuration file.
   c. Copy the downloaded `replica1_data.zip` file to the replica administrative node.

   Note: If a replica administrative node is added after the cluster has been deployed, it will be necessary to update the configuration for each engine node when the replica administrative node is added.

5. Go to **Settings**# **Security**# **Key Pairs** and create a new key pair to assign to the `ADMIN` listener. The key pair needs to be valid for both the administrative and replica administrative nodes.

   Tip: If a replica administrative node is not defined during the initial cluster configuration, you might still opt to define a subject alternative name for a future replica administrative node. This avoids having to reissue the keys when adding the replica administrative node in the future.

6. Open `conf/run.properties` in an editor and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE`.
7. Goto **Settings**# **Networking**# **Listeners** and assign the newly created key pair to the `CONFIG QUERY` listener.

   Note: If `clusterconfig.enabled` is set to `false` in `run.properties`, the key pair needs to be assigned to the `ADMIN` listener instead. This property is set to `false` when a cluster has been upgraded from PingAccess 3.2 or earlier.

8. Restart PingAccess on the administrative node.

Perform steps 9-11 on the replica administrative node, if one has been configured.

9. Unzip `replica1_data.zip` in the *`PA_HOME`* directory.
10. Open `conf/run.properties` in an editor and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE_REPLICA`
11. Start PingAccess on the replica administrative node.

For each engine node, perform steps 12-17.

12. Navigate to **Settings**# **System**# **Clustering** and click **Add Engine**.
13. After defining the engine's parameters, click **Save & Download** to download the engine configuration zip file.
14. Copy *`engine_name`*`_data.zip` to the engine node.
15. On the engine node, unzip *`engine_name`*`_data.zip` in the *`PA_HOME`* directory.
16. On the engine node, open `conf/run.properties` in an editor and change the `pa.operational.mode` value to `CLUSTERED_ENGINE`.
17. Start PingAccess on the engine node.

Results

Go to **Settings**# **System**# **Clustering** to check your cluster's status. If everything is configured properly, the cluster engine nodes and optional replica administrative node should show a green status icon, indicating that the cluster is operational.

You can optionally configure each node to run PingAccess as a service set to automatically run when the node is started. For more information about configuring PingAccess as a service, see the installation documentation.

Configuring the administrative node

You can configure one PingAccess node as the administrative node.

About this task

This procedure allows you to specify an HTTP or HTTPS proxy. If proxy configuration is defined in a properties file (`bootstrap.properties` or `run.properties`), it will take precedence over UI or API configuration.

If a proxy is configured on a replica administrative node, when failing over and before removing the `bootstrap.properties` file, the administrative node should have the same proxy configuration.

Warning: If you are promoting a replica administrative node to an administrative node, remove the bootstrap properties file from the replica administrative node.

Steps

1. Go to **System**# **Clustering**# **Administrative Nodes**.
2. Enter the host and port for the administrative console. The default is `localhost:9000`.
3. If applicable, specify an **HTTP Proxy** for the engine. Click **Create** to create an HTTP proxy.
4. If applicable, specify an **HTTPS Proxy** for the engine. Click **Create** to create an HTTPS proxy.
5. Click **Save**.

Configure runtime state clustering

Runtime state clustering lets multiple engine nodes share information and runtime states. This can improve performance in large environments or environments using rate limiting rules or token mediation.

About this task

Steps

1. Modify *`PA_HOME`*`/conf/run.properties` and change the `pa.cluster.interprocess.communication` value from `none` to either `tcp` or `udp`.

   Using UDP for the interprocess communication allows a multicast group to be used for this communication, which may be more efficient in large environments.
2. If TCP is used for interprocess communication, configure the `pa.cluster.tcp.discovery.initial.hosts` value to specify a list of initial hosts to contact for group discovery.
3. If UDP is used for interprocess communication, optionally configure the `pa.cluster.mcast.group.address` and `pa.cluster.mcast.group.port` values for each group of nodes.
4. Update the `pa.cluster.bind.address` with the IP address of the network interface that should handle the interprocess communication traffic for the cluster.
5. Place a load balancer in front of each group of nodes to distribute the load across the nodes.
6. Restart the engine nodes.

Configuring the replica administrative node

You can configure one PingAccess node as a replica administrative node to provide an alternative if the administrative node fails.

About this task

When using a replica administrative node, you must define a key pair to use for the CONFIG QUERY listener that includes both the administrative node and the replica administrative node. You can do this either by using a wildcard certificate or by defining subject alternative names in the key pair that include the replica administrative node's DNS name. If you use a replica administrative node in your configuration, configure the replica administrative node before defining the engine nodes, or the `bootstrap.properties` files generated for the engine nodes will not include information about the replica administrative node.

In addition to the configuration below, the Replica Administrative node includes a status indicator that indicates the health of the node and a read-only **Last Updated** field that indicates the date and time of the last update. The status indicator can be green (good status), yellow (degraded status), or red (failed status).

The status is determined by using the value for `admin.polling.delay` as an interval to measure health:

Green (good status):

The replica administrative node contacted the primary administrative node on the last pull request.

Yellow (degraded status):

The replica administrative node contacted the primary administrative node between 2 and 10 intervals.

Red (failed status):

The replica administrative node has either never contacted the primary administrative node, or it has been more than 10 intervals since the nodes communicated.

Note: If you are configuring a replica administrative node in the environment, that must be done before you configure the engines.

Steps

1. Go to **System#** **Clustering#** **Administrative Nodes**.
2. Configure the **Host** value. This name and port pair must match either a subject alternative name in the key pair or be considered a match for the wildcard specified if the key pair uses a wildcard in the common name.
3. If applicable, specify an **HTTP Proxy** for the engine. Click **Create** to create an HTTP proxy.
4. If applicable, specify an **HTTPS Proxy** for the engine. Click **Create** to create an HTTPS proxy.
5. Specify the **Replica Administrative Node Trusted Certificate** to use for cases where a TLS-terminating network appliance, such as a load balancer, is placed between the engines and the admin node.
6. Click **Save & Download** to download the *`<replicaname>`*`_data.zip` file for the replica administrative node. PingAccess automatically generates and downloads a public and private key pair into the `bootstrap.properties` file for the node. The **Public Key** is indicated on this screen.
7. Copy the downloaded file to the replica administrative node's *`PA_HOME`* directory and unzip it.
8. If the replica administrative node is running on a Linux host, execute the command **`chmod 400 conf/pa.jwk`**.
9. Edit *`PA_HOME`*`/conf/run.properties` on the replica administrative node and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE_REPLICA` .
10. Start the replica administrative node.
11. Verify replication has completed by monitoring the *`PA_HOME`*`/log/pingaccess.log` file and looking for the message "Configuration successfully synchronized with administrative node".

Manual fail over to the replica administrative node

You can manually promote the replica administrative node to the administrative node if the administrative node has failed.

About this task

The replica administrative node is intended to be used for disaster recovery purposes. If the clustered console is recoverable, then that recovery should be used rather than failing over to the replica administrative node.

Warning: Only one primary administrative node should be running for the cluster at any given time.

Steps

1. Open *PA_HOME*`/conf/run.properties` in an editor.
2. Locate the `pa.operational.mode` line and change the value from `CLUSTERED_CONSOLE_REPLICA` to `CLUSTERED_CONSOLE`.

   This change is detected while the node is running, and does not require a restart of the node.

Reinstate a replica administrative node after failing over

After you have failed over to your replica administrative node, you must configure a new replica administrative node.

About this task

After the console has been failed over to the replica, you need to set up a new replica administrative console.

Note:

If you want to then switch back to the original console, shut down the original replica node, and fail over back to the newly created replica console. Follow these steps a second time to re-establish the original replica node.

Steps

1. Install the new replica administrative node.
2. Change the `run.properties` value for `pa.operational.mode` to `CLUSTERED_CONSOLE_REPLICA`
3. Go to **Settings**# **System**# **Clustering**# **Administrative Nodes** and change the **Primary Administrative Node** hostname and port to the failed-over node.
4. Remove the **Replica Administrative Node** public key, then change the **Replica Administrative Node** hostname and port to point to the new replica node.

   Tip: If your key pair does not include a wildcard, you can use the same hostname as the original console to avoid having to recreate the console key pair and the `bootstrap.properties` files for each engine.
5. Click **Save & Download** to download the bootstrap file for the replica administrative node.
6. Copy the downloaded file to the new replica administrative node's *<PA_HOME>*`/conf` directory, and rename it to `bootstrap.properties`.
7. Edit *PA_HOME*`/conf/run.properties` on the new replica administrative node and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE_REPLICA`.
8. Start the new replica node.
9. Verify replication has completed by monitoring the *PA_HOME*`/log/pingaccess.log` file and looking for the message "Configuration successfully synchronized with administrative node".

Configure an engine node

You can configure one or more engine nodes within your cluster to manage client requests.

About this task

For each engine node:

Steps

1. Click **Settings**# **System**# **Clustering**.
2. Click **Add Engine** to configure a new engine node.
3. Enter a **Name** for the engine node. Special characters and spaces are allowed.
4. Enter a **Description** of the engine node.
5. If applicable, specify an **HTTP Proxy** for the engine node. Click **Create** to create an HTTP proxy.
6. If applicable, specify an **HTTPS Proxy** for the engine node. Click **Create** to create an HTTPS proxy.
7. Specify the **Engine Trusted Certificate** to use for cases where a TLS-terminating network appliance, such as a load balancer, is placed between the engines and the administrative node.
8. Click **Save & Download** to generate and download a public and private key pair into the `<enginename>_data.zip` file for the engine. This file is prepended with the name you give the engine node. Depending on your browser configuration, you may be prompted to save the file.
9. Copy the zip file to the `PA_HOME` directory of the corresponding engine node in the cluster and unzip it. The engine uses these files to authenticate and communicate with the administrative console.

> **Info:** You can generate a new key for an engine node at any time by clicking **Save & Download** and unzipping the `<enginename>_data.zip` archive on the engine node to replace the files with a new set of configuration files. When that engine node starts up and begins using the new files, PingAccess deletes the old key.

10. On Linux engine nodes, change the permissions on the extracted `pa.jwk` to mode 400 by executing the command `chmod 400 conf/pa.jwk` after extracting the zip file.
11. Start each engine node.

Editing an engine node

You can edit the name and description of an engine node within your cluster, and download a new public key if necessary.

Steps

1. Go to **System**# **Clustering**.
2. Expand the node you want to edit, then click ✎.
3. Edit the node **Name** or **Description**, as appropriate.
4. If a new public key is needed, click **Save & Download**.
5. If a new public key is not needed, click **Save**.

Revoke access from an engine node

You can remove an engine node's access to the administrative node.

About this task

If an engine node has been compromised, you can delete its public keys from the administrative node to prevent it from accessing the administrative node. You can recreate these keys after you have recovered the engine node.

Steps

1. Go to **Settings**# **System**# **Clustering**.
2. Expand the engine node you wish to remove from the cluster and edit it.

3. Click **Delete** under the **Public Keys** heading to revoke the engine node's access to the administrative node.

> ⓘ **Info:** You can use the **Save & Download** button to create a new key for the engine. See *Configure an engine node* on page 86 for more information.

4. Click **Save**.

Removing an engine node

You can remove an engine node from the cluster.

Steps

1. Go to **System# Clustering**.
2. Expand the engine node you want to delete and click 🗑 to permanently remove all references to the node from the cluster.
3. Click **Delete** in the confirmation window.

Configuration File Reference Guide

Configuration file reference

This document provides a reference to configurable parameters used by PingAccess at runtime. These parameters are configured in the `run.properties` file located at `<PA_HOME>/conf/`.

> ⓘ **Note:** Changes made to the run.properties file will take effect after PingAccess is restarted.

> ⓘ **Tip:** When storing passwords in `run.properties`, we strongly recommend you obfuscate them using the `obfuscate.bat or obfuscate.sh` utility to mask the password value. This utility is located in the *`PA_HOME`*`/bin` folder.

account.locking.max.consecutive.failures

Defines the maximum number of failed login attempts before locking the account when using basic authentication in the administrative UI or administrative REST APIs. The default value is `3`.

account.locking.max.lockout.period

Defines, in minutes, the amount of time to lock an account out from the administrative interfaces after exceeding the `account.locking.max.consecutive.failures`. The default value is `1`.

admin.acceptors

Defines the number of admin acceptor threads used to establish connections. The default value is `1`.

admin.auth

Overrides the administrator authentication method. For example, if SSO Authentication is enabled and is somehow misconfigured, this property can be used to bypass the database configuration and force the use of Basic Authentication. Default value is `default`.

admin.backlog

Defines the maximum queue length for incoming admin connection indications. The default value is `512`.

admin.bindAddress

Defines the IP address that `admin.port` will bind to. This is typically required on multihomed servers having multiple IP addresses. The default value of `0.0.0.0` means that the port will bind to all of the server's IP addresses.

admin.header.Strict-Transport-Security

Sets the parameters for the Strict-Transport-Security response header sent to the browser when an administrator is interacting with the Admin UI.

admin.header.X-Content-Type-Options

Sets the parameters for the X-Content-Type-Options response header sent to the browser when an admin is interacting with the Admin UI.

admin.header.X-Frame-Options

Sets the parameters for the X-Frame-Options HTTP response header sent to the browser when an admin is interacting with the Admin UI.

admin.header.X-XSS-Protection

Sets the parameters for the X-XSS-Protection HTTP response header sent to the browser when an admin is interacting with the Admin UI.

admin.headers

Additional headers added to responses from the PingAccess Administrator Console and the Administrator API interface. Header values are defined using the `admin.header` prefix.

admin.httptransport.coreThreadPoolSize

Defines the number of threads to keep in the admin transport pool, even if they are idle. The default value is `5`.

admin.httptransport.ioThreads

Defines the number of I/O threads for the admin host. A value of `0` is used to denote that PingAccess should automatically calculate the appropriate number of I/O threads for the host. The default value is `0`.

admin.httptransport.maxThreadPoolSize

Defines the maximum number of threads for the admin transport pool. The default value is `-1`, which denotes no limit.

admin.httptransport.socketTimeout

Defines, in milliseconds, the admin socket timeout. The default value is `30000`.

admin.max.request.bodylength

Defines, in megabytes, the maximum body length for a request to the administrative API endpoint. The default value is `15`.

admin.polling.delay

Defines, in milliseconds, how long after the initial query to the administrative console that the replica administrative node begins querying for configuration information. The default is every `2000` milliseconds.

admin.polling.initialdelay

Defines, in milliseconds, how long after the replica administrative node starts up before it begins to poll the administrative console for configuration information. The default is `500`.

admin.port

Defines the TCP port on which the PingAccess administrative console runs. Default is `9000`.

admin.reuseAddress

When enabled, allows a process to bind to a port which remains in a TIME_WAIT state for the admin transport. The default value is `true`.

admin.ssl.ciphers

Defines the type of cryptographic ciphers available for use with administrative HTTPS ports.

admin.ssl.protocols

Defines the protocols for use with administrative HTTPS ports.

admin.ui.max.sessions

Defines the maximum number of sessions for the admin UI when admin SLO is not enabled.

agent.assets.header.X-Frame-Options

Sets the parameters for the X-Frame-Options HTTP response header sent to the browser via the agent when responding to a request for an asset used by a PingAccess template.

agent.assets.headers

Additional headers added to responses from PingAccess Agents. Header values are defined using the `agent.assets.header` prefix.

agent.authz.header.required

Defines whether PingAccess server should authenticate agent requests using agent name and shared secret in the vnd-pi-authz header. Default value is `true`. Setting this to `false` is useful for POCs and/or debugging.

agent.cache.invalidated.response.duration

Defines the duration in seconds that application configuration changes are sent by PingAccess server to agents using the vnd-pi-cache-invalidated header in agent responses for the changed application. Default value is `900`.

agent.default.token.cache.ttl

Defines, in seconds, the time to live for cached agent tokens.

agent.error.header.X-Frame-Options

Sets the parameters for the X-Frame-Options HTTP response header sent to the browser via the agent when responding with a PingAccess error template.

agent.error.headers

Additional headers added to error responses from PingAccess Agents. Header values are defined using the `agent.error.header` prefix.

agent.http.backlog

Defines the maximum queue length for incoming admin connection indications. The default value is `512`.

agent.http.bindAddress

Defines the address from which an engine listens for agent requests.

agent.http.enabled

Defines whether a STANDALONE or CLUSTERED_ENGINE node listens for agent requests on the port defined by the `agent.http.port` setting. Default is `true`.

agent.http.port

Defines the TCP port on which the engine listens for agent requests. Default is `3030`.

agent.http.reuseAddress

When enabled, allows a process to bind to a port which remains in a TIME_WAIT state for the agent transport. The default value is `true`.

agent.http.secure

Defines whether the engine is using HTTPS for agent requests. Default is `true`.

agent.httptransport.coreThreadPoolSize

Defines the number of threads to keep in the agent transport pool, even if they are idle. The default value is `5`.

agent.httptransport.ioThreads

Defines the number of I/O threads for the agent host. A value of `0` is used to denote that PingAccess should automatically calculate the appropriate number of I/O threads for the host. The default value is `0`.

agent.httptransport.maxThreadPoolSize

Defines the maximum number of threads for the agent transport pool. The default value is `-1`, which denotes no limit.

agent.httptransport.socketTimeout

Defines, in milliseconds, the agent socket timeout. The default value is `30000`.

agent.ssl.ciphers

Defines the type of cryptographic ciphers available for use with agent HTTPS ports.

agent.ssl.protocols

Defines the protocols used for communication with agent HTTPS ports.

as.ssl.ciphers

Defines the type of cryptographic ciphers available for use with authorization server HTTPS ports.

as.ssl.protocols

Defines the protocols used for communication with authorization server HTTPS ports.

client.ioThreads

Defines the number of threads for client connections to backend sites. A value of `0` means there is no limit. The default value is `0`.

clusterconfig.acceptors

Defines the number of cluster configuration acceptor threads used to establish connections. The default value is `1`.

clusterconfig.backlog

Defines the maximum queue length for incoming cluster configuration connection indications. The default value is `512`.

clusterconfig.bindAddress

Defines the optional address used for cluster configuration.

clusterconfig.enabled

When enabled, uses the cluster coonfiguration port for cluster replication. When disabled, the admin port is used for cluster configuration replication. The default value is `true`.

> **Note:** This parameter is set to `false` by the PingAccess Upgrade Utility after a PingAccess cluster is upgraded from a version earlier than 4.0.

clusterconfig.httptransport.coreThreadPoolSize

Defines the number of threads to keep in the cluster configuration transport pool, even if they are idle. The default value is `5`.

clusterconfig.httptransport.ioThreads

Defines the number of I/O threads for the cluster configuration host. A value of `0` is used to denote that PingAccess should automatically calculate the appropriate number of I/O threads for the host. The default value is `0`.

clusterconfig.httptransport.maxThreadPoolSize

Defines the maximum number of threads for the cluster configuration transport pool. The default value is `-1`, which denotes no limit.

clusterconfig.httptransport.socketTimeout

Defines, in milliseconds, the cluster configuration socket timeout. The default value is `30000`.

clusterconfig.port

Defines the optional port used for cluster configuration.

clusterconfig.reuseAddress

When enabled, allows a process to bind to a port which remains in a TIME_WAIT state for the cluster configuration transport. The default value is `true`.

clusterconfig.secure

When enabled, enables SSL communications for the cluster configuration port. The default value is `true`.

clusterconfig.ssl.ciphers

Defines the type of cryptographic ciphers available for use with HTTPS ports in a clustered configuration.

clusterconfig.ssl.protocols

Defines the protocols used for communication with HTTPS ports in a clustered configuration.

enable.detailed.heartbeat.response

When enabled, this setting enables a customizable heartbeat response to be returned. When disabled, the heartbeat endpoint returns a `200 OK` response. The default value is `false`.

engine.admin.configuration.audience

Defines the audience used for cluster authentication. This property must be set to the same value on all nodes in a PingAccess cluster. The default value is `PingAccessAdminServer`.

engine.assets.header.X-Frame-Options

Sets the parameters for the X-Frame-Options HTTP response header sent to the browser via the engine when responding to a request for an asset used by a PingAccess template.

engine.assets.headers

Additional headers added to responses from the PingAccess Engine. Header values are defined using the `engine.assets.header` prefix.

engine.error.header.X-Frame-Options

Sets the parameters for the X-Frame-Options HTTP response header sent to the browser via the engine when responding with a PingAccess error template.

engine.error.headers

Additional headers added to error responses from the PingAccess Engine. Header values are defined using the `engine.error.header` prefix.

engine.http.acceptors

Defines the number of engine acceptor threads used to establish connections. The default value is `1`.

engine.http.backlog

Defines the maximum queue length for incoming engine connection indications. The default value is `512`.

engine.http.bindAddress

Defines the address for an engine in a clustered environment.

engine.http.enabled

Defines whether a STANDALONE or CLUSTERED_ENGINE node listens for requests on the ports defined by the Engine Listeners. Default is `true`.

engine.http.reuseAddress

When enabled, allows a process to bind to a port which remains in a TIME_WAIT state for the engine transport. The default value is `true`.

engine.httptransport.coreThreadPoolSize

Defines the number of threads to keep in the engine transport pool, even if they are idle. The default value is `5`.

engine.httptransport.ioThreads

Defines the number of I/O threads for the engine host. A value of `0` is used to denote that PingAccess should automatically calculate the appropriate number of I/O threads for the host. The default value is `0`.

engine.httptransport.maxThreadPoolSize

Defines the maximum number of threads for the engine transport pool. The default value is `-1`, which denotes no limit.

engine.httptransport.socketTimeout

Defines, in milliseconds, the engine socket timeout. The default value is `30000`.

engine.polling.delay

Defines, in milliseconds, how long after the initial query to the administrative console that the engine begins querying for configuration information. The default is every `2000` milliseconds.

engine.polling.initialdelay

Defines, in milliseconds, how long after the engine starts up before it begins to poll the administrative console for configuration information. The default is `500`.

engine.ssl.ciphers

Defines the type of cryptographic ciphers available for use with engine HTTPS ports.

engine.ssl.protocols

Defines the protocols used with engine HTTPS ports.

engine.websocket.maxConnections

Sets the maximum number of allowed web socket connections. Default is -1 (unlimited).

pa.admin.test.connections

A boolean property that allows the PingAccess admin UI to make HTTP calls to validate that it can reach PingFederate and sites when the user configures them.

pa.admin.user.password.error.message

Defines the message returned when password complexity is not satisfied. The default value is `Password must be at least 8 characters in length, contain one upper-case letter, one lower-case letter and one digit.`.

pa.admin.user.password.regex

Defines the regex that controls password complexity for the Administration Console. The default value is

```
((?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{8,20})
```

pa.auditing.unknown.resource

When set to `true`, this setting causes PingAccess to audit requests for resources that are requested but not mapped to an Application or Resource. This setting can be used to help troubleshoot resource definition issues. The default is `false`.

pa.backup.filesToKeep

Defines the number of backup files to preserve when the Administrator authenticates to PingAccess. The default value is `25`. A value of `0` disables the creation of backup files.

> **Note:** Disabling the creation of backup files can speed up the login process in large environments. If you disable the creation of backup files, use the administrative API backup endpoint to create regular backups.

pa.cluster.auth.pwd

Sets the key that each engine in the cluster must use to authenticate when joining the group. This prevents unauthorized engines from joining a cluster. This key should be treated as a strong key rather than as a human-readable password value. (Values: any string or blank)

> **Important:** If `pa.cluster.encrypt` is true, `pa.cluster.auth.pwd` must not be blank.

pa.cluster.bind.address

Defines the IP address to which you bind the TCP or UDP listener. The default is `127.0.0.1`.

pa.cluster.bind.port

The port associated with the bind-address property above. The default is `7610`. Whether this is a TCP or UPD port depends on the value configured for the `pa.cluster.interprocess.communication` property (see above).

pa.cluster.encrypt

Indicates whether to encrypt network traffic sent between engines in a cluster. (Values: `true` or `false` [default])

> **Important:** If `pa.cluster.encrypt` is true, `pa.cluster.auth.pwd` must not be blank.

pa.cluster.failure.detection.bind.port

Indicates the bind port of a server socket that is opened on the given engine and used by other engines as part of one of the cluster's failure-detection mechanisms. This port is bound to the address determined by `pa.cluster.bind.address`. The default is `7710`. Whether this is a TCP or UDP port depends on the value configured for the `pa.cluster.interprocess.communication` property (see above).

pa.cluster.interprocess.communication

Defines how the JGroups cluster communicates. `none` (the default): Indicates that no communication is configured between servers in the cluster. `udp`: Indicates that the cluster uses Multicast communications to send and receive information to and from multiple servers at once. `tcp`: Indicates that the cluster uses Unicast communications to send and receive information to and from individual servers one at a time.

pa.cluster.mcast.group.address

Defines the IP address shared among engines in the same cluster for UDP multicast communication; required when the interprocess communication mode is set to `udp`. (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 engines in a cluster must use the same address for this property and the port property below. The default value is `239.16.96.69`.

pa.cluster.mcast.group.port

Defines the UDP port associated with the `pa.cluster.mcast.group.address` property above. The default value is `7611`.

pa.cluster.serverstate.replicationIntervalMilliseconds

Defines, in milliseconds, how often Rate Limiting metadata is replicated within a subcluster. The default value is `1000`.

pa.cluster.serverstate.staleEntryEvictionIntervalSeconds

Defines, in seconds, how often a PingAccess engine scans the Rate Limiting metadata to evaluate metadata to be removed from the cache, based on the `pa.cluster.serverstate.timeToIdleSeconds` value. The default value is `60`.

pa.cluster.serverstate.timeToIdleSeconds

Defines, in seconds, how long metadata for the Rate Limiting rule is maintained by a PingAccess Engine after its last use. The default value is `86400`.

pa.cluster.tcp.discovery.initial.hosts

Designates the initial hosts to be contacted for group membership information when discovering and joining the group; required when the interprocess communication mode is set to `tcp`. The value is a comma-separated list of host names (or IP addresses) and ports. For example, `127.0.0.1[7602]`.

pa.default.availability.ondemand.connectTimeout

Defines, in milliseconds, the amount of time to wait before trying to connect to the remote host. The default is `10000`.

pa.default.availability.ondemand.failedRetryTimeout

Defines, in seconds, the amount of time to wait before retrying a failed host. The default is `60`.

pa.default.availability.ondemand.maxRetries

Defines the maximum number of retries before marking the target system down. The default is `2`.

pa.default.availability.ondemand.pooledConnectionTimeout

Defines, in milliseconds, the amount of time to wait before timing out the request for a pooled connection to the target site. The default is `-1`.

pa.default.availability.ondemand.readTimeout

Defines, in milliseconds, the amount of time to wait before timing out the read response for a target site. The default is `-1`.

pa.default.availability.ondemand.retryDelay

Defines, in milliseconds, the amount of time to wait after a timeout before retrying the host. The default is `250`.

pa.default.contentRewrite.buffer.default

Defines, in bytes, the default buffer size when using a Rewrite Content rule to do a search and replace of content. The default value is `2048`.

pa.default.contentRewrite.buffer.min

Defines, in bytes, the minimum buffer size used when using a Rewrite content rule. The default value is `1024`.

pa.default.limitRequestLine

Defines the maximum number of bytes to read from the request line. The default value is `8192`.

pa.default.maxConnectionsPerSite

Defines the maximum number of connections PingAccess will open to the PingFederate Admin or Engine. A value of `-1` means there is no limit. The default is `-1`.

pa.default.maxHeaderCount

Defines the maximum number of headers to read from a request. The default value is `100`.

pa.default.maxHttpHeaderSize

Defines the maximum number of bytes to read when reading headers. The default value is `8192`.

pa.default.maxRequestBodySize

Defines the maximum number of bytes to read from a request body. The default value is `204800`.

pa.default.session.cookie.attributes.httponly

Defines the default setting for the **HTTP-Only Cookie** setting for newly-created web sessions. The default value is `true`.

pa.default.session.cookie.attributes.secure

Defines the default setting for the **Secure Cookie** setting for newly-created web sessions. The default value is `true`.

pa.default.session.cookie.size.threshold

Defines, in bytes, the default maximum session cookie size. The default value is `4093`.

pa.ehcache.AuthTokenCache.maxEntriesLocalHeap

Defines the maximum size of the JWT identity mapping token cache used when sending tokens to a protected site. Default is 10000.

pa.ehcache.PATokenValidationCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for decryption of signed or encrypted PingAccess tokens. The default is `10000`.

pa.ehcache.PATokenValidationCache.timeToIdleSeconds

Defines, in seconds, the time an entry in the token validation cache can be idle before it is expired. The default is `120` seconds.

pa.ehcache.PATokenValidationCache.timeToLiveSeconds

Defines, in seconds, the maximum time an entry can be in the token validation cache. The default is `300` seconds.

pa.ehcache.PAWamUserAttributesCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for the PA WAM user attribute cache. The default is `10000`.

pa.ehcache.PAWamUserAttributesCache.timeToIdleSeconds

Defines, in seconds, the time an entry in the PA WAM user attribute cache can be idle before it is expired. The default is `120` seconds.

pa.ehcache.PAWamUserAttributesCache.timeToLiveSeconds

Defines, in seconds, the maximum time an entry can be in the PA WAM user attribute cache. The default is `300` seconds.

pa.ehcache.PFSessionValidationCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for the session validation cache. The default is `10000`.

pa.ehcache.PFSessionValidationCache.timeToIdleSeconds

Defines, in seconds, the time an entry in the session validation cache can be idle before it is expired. The default is `120` seconds.

pa.ehcache.PFSessionValidationCache.timeToLiveSeconds

Defines, in seconds, the maximum time an entry can be in the session validation cache. The default is `300` seconds.

pa.ehcache.PingFederateReferenceTokenCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for OAuth tokens. The default is `10000`.

pa.ehcache.ServiceTokenCache.maxEntriesLocalHeap

Defines the maximum number of entries in the local heap for token mediation. The default is `10000`.

pa.ehcache.ServiceTokenCache.timeToIdleSeconds

Defines, in seconds, the time an entry in the token mediation cache can be idle before it is expired. The default is `1800` seconds.

pa.ehcache.ServiceTokenCache.timeToLiveSeconds

Defines, in seconds, the maximum time an entry can be in the token mediation cache. The default is `14400` seconds.

pa.ehcache.SessionStateCache.maxEntriesLocalHeap

Defines the maximum size of the identity attribute entry cache when the user's attributes are stored on the server rather than as a cookie. Default is 10000.

pa.interceptors.relativepath.decode.count

Number of times the URL is decoded to check for path traversal characters. The default is `3`.

pa.interceptors.relativepath.decode.regex

Defines the accepted URL regex pattern that administrators can customize based on their needs. The default value is:Defines the regular expression to use when checking for a valid path in an incoming request. The default value is

```
[\\p{Po}\\p{N}\\p{Z}\\p{L}\\p{M}\\p{Zs}\\./_\\-\\\\~()\\{\\}\\[\
\]]*
```

Note: This value is double-escaped as required by the `java.util.regex.Pattern` Java class.

pa.interceptors.relativepath.strict

When this property is set to `true`, the incoming URL is matched with the whitelist pattern defined in `pa.interceptors.relativepath.decode.regex`. All other request URLs are rejected. The default value is `false`.

pa.jdbc.filepassword

Defines the password used to encrypt the PingAccess configuration database. Default is `2Access`.

pa.jdbc.password

Defines the password for the database user of the PingAccess configuration database. Default is `2Access`.

pa.jdbc.username

Defines the username for accessing the PingAccess configuration database. Default is `sa`.

pa.keystore.pw

Defines the password for the `$JAVA_HOME/lib/security/cacerts` keystore.

pa.localization.missing.message.placeholder

Defines the message used when an error message is unresolvable. An error will be logged.

pa.localization.resource.bundle.cache.enable

When set to `false`, allows language files in /conf/localization to be added or modified. When `true`, enables caching of language files and properties.

pa.oidc.logout.redirect

A boolean property that defines whether or not to redirect the user to the token provider on logout. If `true`, the user is redirected to the token provider.

pa.oidc.logout.redirectURI

The URI that a user gets sent to when they log out.

pa.oidc.post.preservation.encrypt

When enabled, POST data preserved through a redirection to PingFederate for authentication is encrypted on the client to be used after the authentication is successful. The default value is `false`.

pa.oidc.post.preservation.maxRequestBodySize

Defines, in bytes, the maximum size of the post body for POST preservation. The default value is `8192`.

pa.oidc.post.preservation.paramsAttributeName

Used to store the encoded or encrypted POST payload in the browser session storage during POST preservation.

pa.operational.mode

Controls the operational mode of the PingAccess server in a cluster. Valid values are:

- STANDALONE - Use this value for a standalone (unclustered) PingAccess instance that runs both the administrative console and the engine. This is the default.
- CLUSTERED_CONSOLE - Use this value for the server instance you want to use as the administrative console server.

Info: Only one engine in a cluster can run the administrative console.

- CLUSTERED_CONSOLE_REPLICA - Use this value for the server instance you want to use as the backup administrative console server.
- CLUSTERED_ENGINE - Use this value to indicate a server engine.

Note:

Define the following Engine and Admin properties depending on what operational mode an engine is using.

- Define all of the following Engine and Admin properties when `pa.operational.mode` is set to `STANDALONE`.
- Define only the Admin properties when using `CLUSTERED_CONSOLE` or `CLUSTERED_CONSOLE_REPLICA` mode.
- Define only the Engine properties when using `CLUSTERED_ENGINE` mode.

pa.uri.strict

When enabled, this setting requires the raw input URI be in strict compliance with the URI spec implemented by `java.net.URI` when generating URIs. The default value is `false`.

pa.websession.cookie.sameSiteExcludedUserAgentPatterns

A comma-separated list of regex that specifies whether an end-user browser should have `SameSite=None` applied to cookies issued to it. If the user-agent header from a request matches any of the values in the list, any PA-issued cookie is set with no SameSite attribute if `SameSite=None` would otherwise have been applied. The default value is:

```
^.*\\(iP.+; CPU .*OS 12[_\\d]*.*\\) AppleWebKit\\/.*$,\
^.*Macintosh;.*Mac OS X 10_14.*Version.*Safari.*$,\
^.*(Chromium|Chrome)\\/(5[1-9]|6[0-6])\\.(\\d+)(?:\\.(\\d+)|)(?:\
\.(\\d+)|).*$,\
^.*UCBrowser\\/[0-9][0-1]?.(\\d+)\\.(\\d+)[\\.\\d]*.*$,\
^.*UCBrowser\\/12.[0-9][0-2]?.(\\d+)[\\.\\d]*.*$,\
^.*UCBrowser\\/12.13.[0-2][\\.\\d]*.*$
```

pf.api.keepAliveTimeout

Defines, in milliseconds, the keep alive timeout for the PingFederate API. The default value is `30000`.

pf.api.maxConnections

Defines the maximum number of connections PingAccess will establish to the PingFederate API endpoint. A value of `-1` means there is no limit. The default value is `-1`.

pf.api.maxRetries

Defines the maximum number of retries PingAccess attempts to make to the PingFederate server before delcaring the server unavailable. The default value is `0`.

pf.api.readTimeout

Defines, in milliseconds, how long the API will wait for responses from PingFederate when making calls to the PingFederate Admin API. The default value is `-1`.

pf.api.socketTimeout

Defines, in milliseconds, the socket timeout for the PingFederate API endpoint. The default value is `5000`.

pf.redirect.header.X-Frame-Options

Sets the parameters for the X-Frame-Options value that is sent when the user is redirected to PingFederate to authenticate.

pf.redirect.headers

Additional headers added to the redirection response that sends the client to PingFederate for authentication. Header values are defined using the `pf.redirect.header` prefix.

pf.ssl.ciphers

Defines the type of cryptographic ciphers available for use with PingFederate HTTPS ports.

pf.ssl.protocols

Defines the protocols used for communication with PingFederate HTTPS ports.

provider.ssl.ciphers

Defines the type of cryptographic ciphers available for use with Provider HTTPS ports.

provider.ssl.protocols

Defines the protocols used for communication with Provider HTTPS ports.

rule.error.headers

Additional headers added to responses that result from policy rule results. Header values are defined using the `rule.error.header` prefix.

site.ssl.ciphers

Defines the type of cryptographic ciphers available for use with Site HTTPS ports.

site.ssl.protocols

Defines the protocols used for communication with Site HTTPS ports.

tls.default.cipherSuites

Defines the default set of ciphers used for HTTPS communication.

Note: Legacy browsers may require the addition of SHA1-based ciphers to negotiate a cipher suite with the server. In this case, add the following ciphers to the run.properties file and restart PingAccess:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA

tls.default.protocols

Defines the default protocols used for HTTPS communication.

Deployment Reference Guide

PingAccess deployment guide

There are many topics to consider when deciding how PingAccess fits into your existing network, from determining the deployment architecture required for your use case and whether high-availability options are required.

This section provides information to help you make the right decisions for your environment.

Use cases and deployment architecture

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

For example, you can design a deployment that supports mobile and API access management, Web access management, or auditing and proxying. For each of these environments, you can choose a stand-alone deployment for proof of concept or deploy multiple PingAccess servers in a cluster configuration for high availability, server redundancy, and failover recovery.

You have a choice between using PingAccess as a Gateway or using a PingAccess Agent plugin on the web server. In a gateway deployment, all client requests first go through PingAccess and are checked for authorization before they are forwarded to the target site. In an agent deployment, client requests go directly to the web server serving up the target site, where they are intercepted by the Agent plugin and checked for authorization before they are forwarded to the target resource. The same access control checks are performed by the PingAccess Policy Server in both cases and only properly authorized client request are allowed to reach the target assets. The difference is that in a gateway deployment client requests are rerouted through PingAccess Gateway, while in an agent deployment they continue to be routed directly to the target site, where PingAccess Agent is deployed to intercept them.

PingAccess Agent makes a separate access control request to PingAccess Policy Server using the PingAccess Agent Protocol (PAAP). The *agent request* contains just the relevant parts of the client request so that PingAccess Policy Server can make the access control decision and respond with instructions to the agent regarding any modifications to the original client request that the agent should perform prior to forwarding the request. For example, the agent may add headers and tokens required by the target resource. Under the PingAccess Policy Server's control, the agent may perform a certain amount of caching of information in order to minimize the overhead of contacting the PingAccess Policy Server, thus minimizing response time.

In both gateway and agent deployment the response from the target resource is processed on the way to the original client. In an agent deployment, the amount of processing is more limited than in a gateway deployment. The agent does not make another request to the Policy Server, so response processing is based on the initial agent response. Consequently, the agent is not able to apply the request processing rules available to the gateway.

When designing a deployment architecture, many requirements and components must be identified for a successful implementation. Proper network configuration of routers/firewalls and DNS ensure that all traffic is routed through PingAccess for the Resources it is protecting and that alternative paths (for example, backdoors) are not available.

The following sections provide specific use cases and deployment architecture requirements to assist with designing and implementing your PingAccess environment.

Deploy for Gateway Web Access Management

A PingAccess Web access management (WAM) deployment enables an organization to quickly set up an environment that provides a secure method of managing access rights to Web-based applications while integrating with existing identity management infrastructure.

With growing numbers of internal and external users, and more and more enterprise resources available online, it is important to ensure that qualified users can access only those applications to which they have permission. A WAM environment provides authentication and policy-based access management while integrating with existing infrastructure.

Deployed at the perimeter of a protected network between browsers and protected Web-based applications, PingAccess Gateway performs the following actions:

- Receives inbound calls requesting access to Web applications. Web Session protected requests contain a previously-obtained PA token in a cookie derived from the user's profile during an OpenID Connect based login at PingFederate.
- Evaluates application and resource-level policies and validates the tokens in conjunction with an OpenID Connect Policy configured within PingFederate.
- Acquires the appropriate target security token (Site Authenticators) from the PingFederate STS or from a cache (including attributes and authorized scopes) should a Web application require identity mediation.
- Makes authorized requests to the sites where the Web applications reside and responses are received and processed.
- Relays the responses on to the browsers.

The following sections describe sample Proof of Concept and Production architectures for a WAM use case deployment.

- *WAM Gateway POC Deployment Architecture*
- *WAM Gateway Production Deployment Architecture*

Deploy for Agent Web Access Management

A PingAccess Web access management (WAM) agent deployment enables an organization to quickly set up an environment that provides a secure method of managing access rights to Web-based applications while integrating with existing identity management infrastructure and minimal network configuration changes.

With growing numbers of internal and external users, and more and more enterprise resources available online, it is important to ensure that qualified users can access only those applications to which they have permission. A WAM environment provides authentication and policy-based access management while integrating with existing infrastructure.

The PingAccess Agent plugin is installed on the Web server hosting the protected Web-based applications and configured to communicate with PingAccess Server also deployed on the network. When the agent intercepts a client request to a protected Web application resource it performs the following actions:

- Intercepts inbound requests to Web applications.
- Sends agent requests to the PingAccess Policy Server sending along relevant request information needed by Policy Server.
- Receives agent responses from Policy Server and follows the instructions from Policy Server, modifies the request as specified, and allows the request to proceed to the target resource.
- Intercepts responses from the application and modifies response headers as instructed in the initial agent request to Policy Server.
- Relays responses on to the browsers.

The PingAccess Policy Server listens for agent requests and performs the following actions:

- Evaluates application and resource-level policies and validates the tokens in conjunction with an OpenID Connect Policy configured within PingFederate
- Acquires the appropriate HTTP request header configuration from the associated Identity Mappings.
- Sends an agent response with instructions on whether to allow the request and how to modify the client request headers.

The following sections describe sample Proof of Concept and Production architectures for a WAM use case deployment.

- *WAM Agent POC Deployment Architecture*
- *WAM Agent Production Deployment Architecture*

Deploy for Gateway API Access Management

A PingAccess API access management deployment enables an organization to quickly set up an environment that provides a secure method of controlling access to APIs while integrating with existing identity management infrastructure.

Pressure from an ever-expanding mobile device and API economy can lead developers to hastily design and expose APIs outside the network perimeter. Standardized API access management leads to a more consistent, centrally-controlled model that ensures existing infrastructure and security policies are followed, thereby safeguarding an organization's assets.

PingAccess Gateway sits at the perimeter of a protected network between mobile, in-browser, or server-based client applications and protected APIs and performs the following actions:

- Receives inbound API calls requesting protected applications. OAuth-protected API calls contain previously-obtained access tokens retrieved from PingFederate acting as an OAuth Authorization Server.
- Evaluates application and resource-level policies and validates access tokens in conjunction with PingFederate.
- Acquires the appropriate target site security token (Site Authenticators) from the PingFederate STS or from a cache (including attributes and authorized scopes) should an API require identity mediation.
- Makes authorized requests to the APIs and responses are received and processed.
- Relays the responses on to the clients.

The following sections describe sample Proof of Concept and Production architectures for an API access management use case deployment.

- *API Access Management POC Deployment Architecture*
- *API Access Management Production Deployment Architecture*

Deploy for auditing and proxying

A PingAccess deployment for auditing and proxying enables an organization to quickly set up an environment that provides a secure method of controlling access to back-end sites.

With growing numbers of internal and external users, it is important to know which users are accessing applications, from where and when they are accessing them, and ensuring that they are correctly accessing only those applications to which they have permission. A standardized auditing/proxying deployment provides a centrally-controlled model that ensures existing infrastructure and security policies are followed, thereby safeguarding an organization's assets.

Sitting at the perimeter of a protected network between mobile, in-browser, or server-based client applications and back-end Sites, PingAccess performs the following actions:

- Receives inbound calls requesting access to protected back-end Sites.
- Audits the request and then makes authorized requests to the back-end Sites.
- Receives and processes responses and relays them on to the clients.

The following sections describe sample Proof of Concept and Production architectures for an auditing/ proxying use case deployment.

- *Audit and Proxy POC Deployment Architecture*
- *Audit and Proxy Production Deployment Architecture*

Configuration by use case

Your configuration steps will vary depending on what type of deployment you are implementing.

See the *Deployment Guide* for a detailed discussion of deployment considerations and best practices in designing your architecture. The following sections describe the configuration steps for the most common use cases:

- *API Access Management Gateway Deployment*
- *Web Access Management Agent Deployment*
- *Web Access Management Gateway Deployment*
- *Auditing and Proxying Gateway Deployment*

Next steps

Once you complete the above configuration settings, your next steps are similar for all use cases:

- Configure Sites and Agents to define the target applications to be protected. Sites may need Site Authenticators to define the credentials the site expects for access control.
- Configure Applications and Resources to define the assets you wish to allow clients to access.
- Create Policies for the defined applications and resources to protect them.

Web Access Management Gateway deployment

This table describes the important configuration options for a Web Access Management Gateway deployment.

See *Deploying for Gateway Web Access Management* in the *Deployment Guide* for specific use case information.

Step	Description
Configure the connection to the PingFederate.	PingAccess uses PingFederate to manage web session and authentication.
Configure the OpenID Connect Relying Party Client for PingAccess.	The client must be registered with PingFederate and the client credentials configured in PingAccess to identify PingAccess when requesting authentication for users trying to access Web applications.

Step	**Description**
Configure Web session details to enable protection of Web Resources.	Configures settings for secure Web sessions such as timeout values, cookie parameters, and cryptographic algorithms.
Generate or Import Key Pairs and configure HTTP Listeners.	Defines the certificates and keys used to secure access to the PingAccess administrative console and secure incoming HTTPS requests at runtime.
Set up your cluster for high availability.	Facilitates high availability of critical services, and increases performance and overall system throughput.
Add trusted CA certificates.	Defines trust to certificates presented during outbound secure HTTPS connections.
Create a trusted certificate group.	Provides a trusted set of anchor certificates for use when authenticating outbound secure HTTPS connections.
Define virtual servers for protected resources.	Allows one server to share PingAccess Resources without requiring all Sites on the server to use the same host name. If SNI is available (Java 8), specific key pairs can be assigned to virtual hosts.

Web Access Management Agent deployment

This table describes the important configuration options for a Web Access Management Agent deployment.

See *Deploying for Agent Web Access Management* for specific use case information.

First, PingAccess Agent needs to be deployed using the following steps:

1. Install PA Agent on Web Server - following instruction in *PingAccess Agent for Apache Installation* or *PingAccess Agent for IIS Installation* depending on your specific Web server.
2. Define the Agents and download agent bootstrap.properties file via the download field in the Shared Secrets field.
3. Deploy the agent bootstrap.properties file to agents following instructions in *PingAccess Agent Configuration* .

The rest of PingAccess deployment is similar to *Web Access Management Gateway Deployment*.

Step	**Description**
Configure the connection to the PingFederate.	PingAccess uses PingFederate to manage web session and authentication.
Configure the OpenID Connect Relying Party Client for PingAccess.	The client must be registered with PingFederate and the client credentials configured in PingAccess to identify PingAccess when requesting authentication for users trying to access Web applications.
Configure Web session details to enable protection of Web Resources.	Configures settings for secure Web sessions such as timeout values, cookie parameters, and cryptographic algorithms.
Generate or Import Key Pairs and configure HTTP Listeners.	Defines the certificates and keys used to secure access to the PingAccess administrative console and secure incoming HTTPS requests at runtime.

Step	Description
Set up your cluster for high availability.	Facilitates high availability of critical services, and increases performance and overall system throughput.
Add trusted CA certificates.	Defines trust to certificates presented during outbound secure HTTPS connections.
Create a trusted certificate group.	Provides a trusted set of anchor certificates for use when authenticating outbound secure HTTPS connections.
Define virtual servers for protected resources.	Allows one server to share PingAccess Resources without requiring all Sites on the server to use the same host name. If SNI is available (Java 8), specific key pairs can be assigned to virtual hosts.

API Access Management Gateway deployment

This table describes the important configuration options for deploying an API Gateway.

See *Deploying for Gateway API Access Management* in the *Deployment Guide* for specific use case information.

Step	Description
Configure the connection to the PingFederate OAuth Authorization Server.	PingAccess uses this connection and credentials to validate incoming Access Tokens for securing API calls.
Configure the OpenID Connect Relying Party Client for PingAccess.	The client must be registered with PingFederate and the client credentials configured in PingAccess to authenticate PingAccess when validating incoming Access Tokens.
Generate or Import Key Pairs and configure HTTP Listeners.	Defines the certificates and keys used to secure access to the PingAccess administrative console and secure incoming HTTPS requests at runtime.
Set up your cluster for high availability.	Facilitates high availability of critical services, and increases performance and overall system throughput.
Add trusted CA certificates.	Defines trust to certificates presented during outbound secure HTTPS connections.
Create a trusted certificate group.	Provides a trusted set of anchor certificates for use when authenticating outbound secure HTTPS connections.
Define virtual servers for protected applications.	Allows one server to share PingAccess Resources without requiring all Sites on the server to use the same host name. If SNI is available (Java 8), specific key pairs can be assigned to virtual hosts

Auditing and proxying Gateway deployment

This table describes the important configuration options for an auditing or proxying deployment.

See *Deploying for Auditing and Proxying* for specific use case information.

Step	Description
Generate or Import Key Pairs and configure HTTP Listeners.	Defines the certificates and keys used to secure access to the PingAccess administrative console and secure incoming HTTPS requests at runtime.
Set up your cluster for high availability.	Facilitates high availability of critical services, and increases performance and overall system throughput.
Add trusted CA certificates.	Defines trust to certificates presented during outbound secure HTTPS connections.
Create a trusted certificate group.	Provides a trusted set of anchor certificates for use when authenticating outbound secure HTTPS connections.
Define virtual servers for protected resources.	Allows one server to share PingAccess Resources without requiring all Sites on the server to use the same host name.

Web Access Management

With growing numbers of internal and external users, and more and more enterprise resources available online, it is important to ensure that qualified users can access only those resources to which they have permission. PingAccess uses Web Access Management (WAM) capabilities to allow organizations to manage access rights to Web-based resources.

WAM is a form of identity management that controls access to Web resources, providing authentication and policy-based access management. Once a user is authenticated, PingAccess applies application and resource-level policies to the request. Once policy evaluation is passed, any required identity mediation between the back-end site and the authenticated user is performed. The user is then granted access to the requested resource.

PingAccess provides two deployment architectures for Web Access Management - gateway and agent. In a gateway deployment client requests are routed to PingAccess which then forwards authorized requests to the target application. In an agent deployment, client requests are intercepted at the web server hosting the application via the PingAccess agent plugin. The agent then communicates with PingAccess Policy Server to validate access before allowing the request to proceed to the target application resource.

Choose Between an Agent or Gateway deployment

PingAccess can be deployed using Agents, as a Gateway (or reverse proxy), or using a combination of both. Before deciding on a deployment, it is important to understand the pros and cons of each deployment scenario and determine how they impact your strategy.

Gateway

Pros:

- Fewer number of deployed components that require maintenance
- Independent of target application platform
- No impact on web/app server processing and performance
- Able to work with existing security token types (such as creating 3rd party WAM tokens)

Cons:

- Requires networking changes
- Requires strategy for securing direct access to backend web/app servers (network routing or service level authentication)
- Depending on the application, may require content / request/response rewriting
- Another layer that requires HA/DR planning

Agents

Pros:

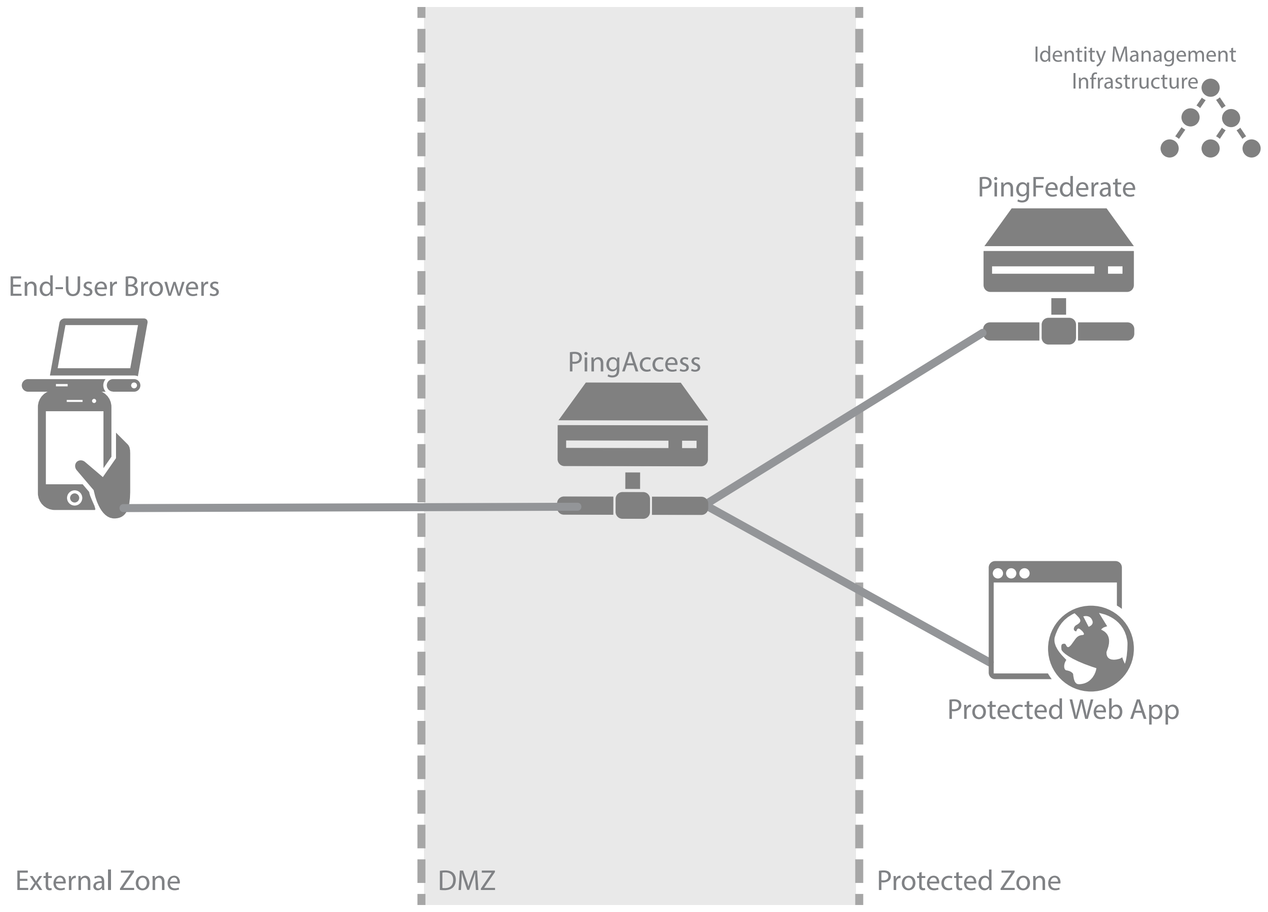
- No networking or server level authentication changes required
- Tight integration with web server handling requests
- Scales with application

Cons:

- High cost of ownership when many agent instances deployed, although should be upgradable/ patchable independently of PingAccess (policy) server
- Policy evaluation is cached; although it is periodically flushed/re-evaluated (for new sessions, updates to session token, etc.) it isn't quite is "real time" as proxy
- Tight dependency on web server version & platform

Web Access Management Gateway proof of concept deployment architecture
This environment is used to emulate a WAM gateway production environment for testing purposes.

In the test environment, PingAccess can be set up with the minimum hardware requirements. This environment example does not provide high availability and is not recommended for a Production environment.



The following table describes the three zones within this proposed architecture.

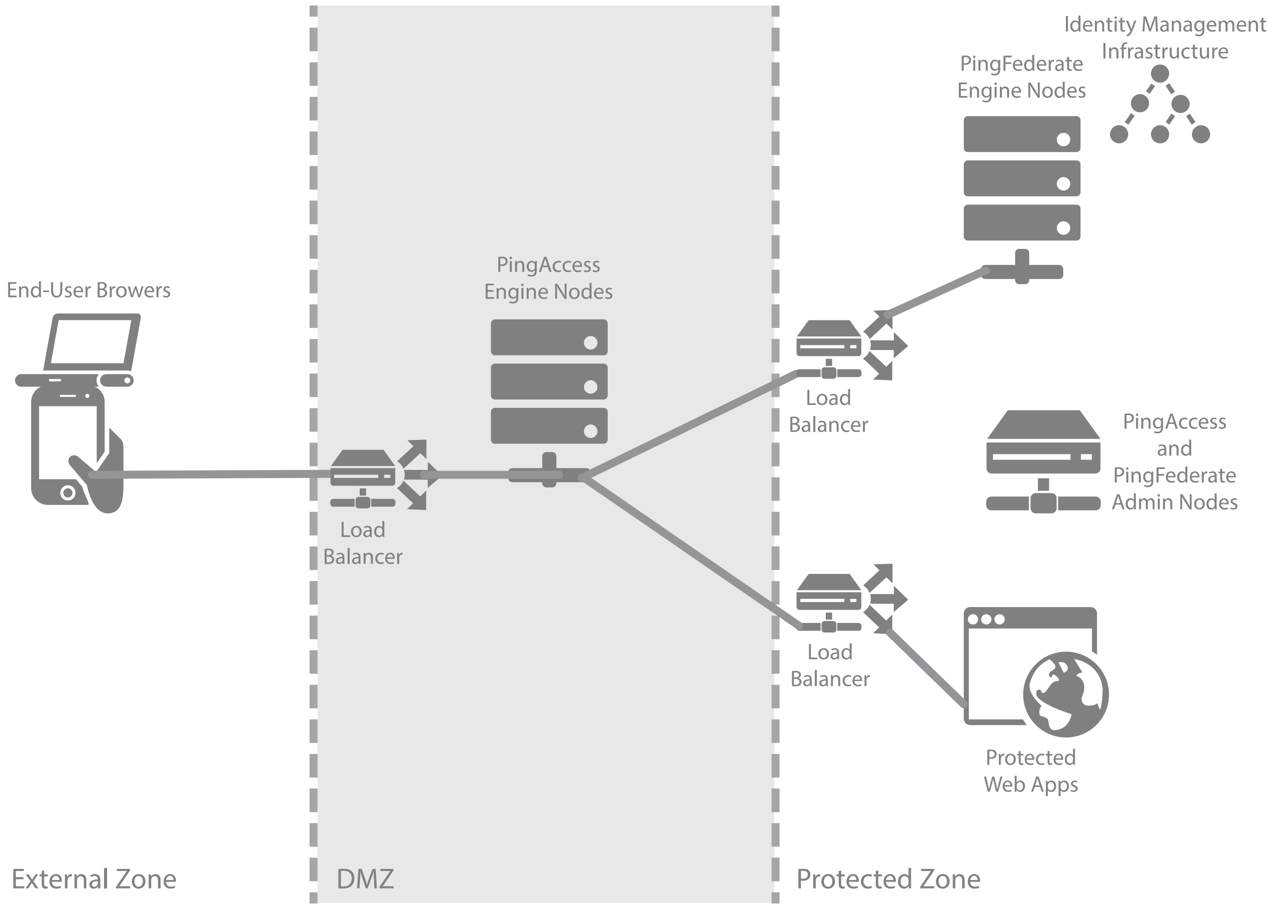
Zone	Description
External Zone	External network where incoming requests for Web applications originate.

Zone	Description
DMZ	Externally exposing segment where PingAccess is accessible to Web browsers. PingAccess is a standalone instance in this environment, serving as both a runtime and an administrative port.
Protected Zone	Back-end controlled zone in which Sites hosting the protected Web applications are located. All requests to these Web applications must be designed to pass through PingAccess. PingFederate is accessible to Web browsers in this zone and is a standalone instance in this environment, serving as both a runtime and an administrative port. PingFederate requires access to identity management infrastructure in order to authenticate users (depicted by the icon in the diagram).

Web Access Management Gateway production deployment architecture
This environment shows a WAM gateway production architecture.

There are many considerations when deploying a Production environment. For high availability and redundancy, the environment requires clustering and load-balancing. Load balancers are required as part of the networking infrastructure to achieve high availability by ensuring that requests are sent to available servers they are front-ending. Best practices in network design and security also include firewalls to ensure that only required ports and protocols are permitted across zones.

Info: PingAccess can provide high availability and basic load balancing for the protected web apps in the protected zone. See the Availability Profiles and Load Balancing Strategies documentation for more information.



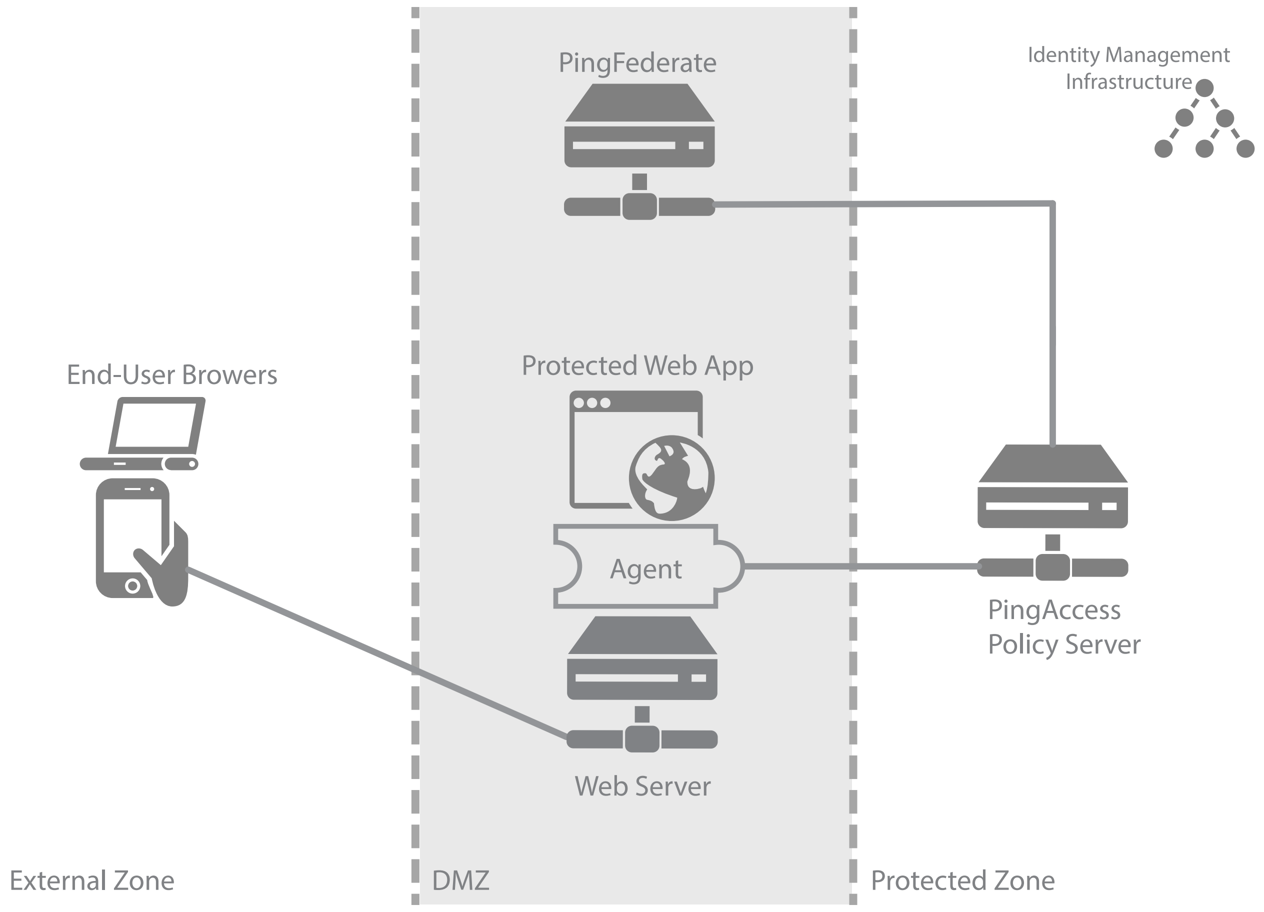
The following table describes the three zones within this proposed architecture.

Zone	Description
External Zone	External network where incoming requests for Web applications originate.
DMZ	Externally exposing segment where PingAccess is accessible to Web browsers. A minimum of two PingAccess engine nodes will be deployed in the DMZ to achieve high availability. Depending on your scalability requirements, more nodes may be required.
Protected Zone	Back-end controlled zone in which Sites hosting the protected Web applications are located. All requests to these Web applications must be designed to pass through PingAccess. PingFederate is accessible to Web browsers in this zone and requires access to identity management infrastructure in order to authenticate users (depicted by the icon in the diagram). A minimum of two PingFederate engine nodes will be deployed in the protected zone. Administrative nodes for both PingAccess and PingFederate may be co-located on a single machine to reduce hardware requirements.

Web Access Management Agent proof of concept deployment architecture

This environment is used to emulate a WAM agent production environment for testing purposes.

In the test environment, PingAccess can be set up with the minimum hardware requirements. This environment example does not provide high availability and is not recommended for a Production environment.



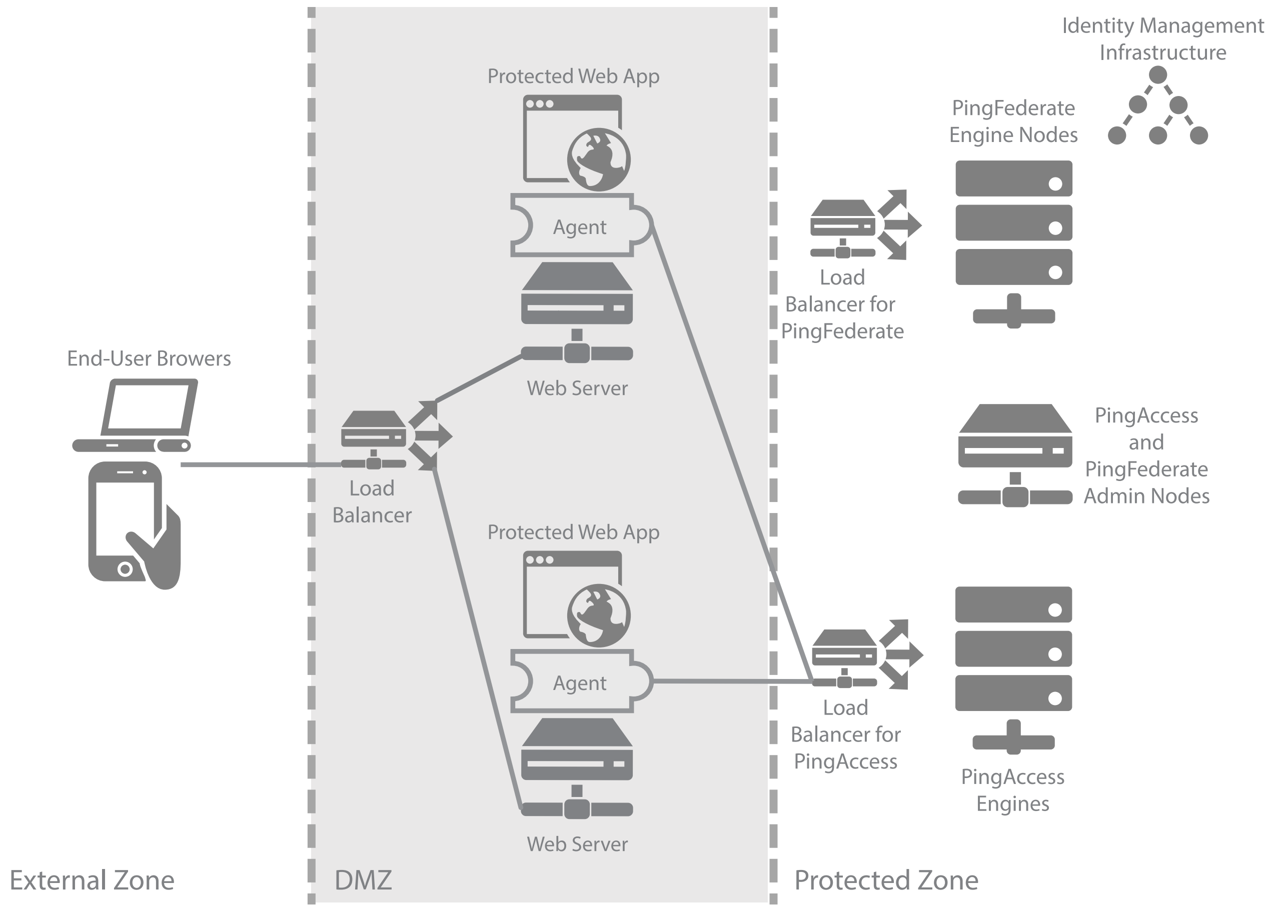
The following table describes the three zones within this proposed architecture.

Zone	Description
External Zone	External network where incoming requests for Web applications originate.
DMZ	Externally exposed segment where application Web server is accessible to Web clients. PingAccess Agent is deployed as a plugin on this Web server. The agent interacts with PingAccess Policy Server in the Protected Zone. PingFederate is deployed as a standalone instance in this environment because during user authentication clients interact with PingFederate. PingFederate requires access to Identity Management Infrastructure in order to authenticate users.

Zone	Description
Protected Zone	Back-end controlled zone with no direct access by Web clients. PingAccess Policy Server is deployed in this zone. PingAccess interacts with PingFederate in the DMZ Zone. Identity Management Infrastructure is deployed in this zone.

Web Access Management Agent production deployment architecture
This environment shows a WAM agent production architecture.

There are many considerations when deploying a Production environment. For high availability and redundancy, the environment requires clustering and load-balancing. Load balancers are required as part of the networking infrastructure to achieve high availability by ensuring that requests are sent to available servers they are front-ending. Best practices in network design and security also include firewalls to ensure that only required ports and protocols are permitted across zones.


Identity Management Infrastructure
Protected Web App
PingFederate Engine Nodes
Agent
Load Balancer for PingFederate
End-User Browers
Web Server
PingAccess and PingFederate Admin Nodes
Load Balancer
Protected Web App
Agent
Load Balancer for PingAccess
PingAccess Engines
Web Server
External Zone
DMZ
Protected Zone

The following table describes the three zones within this proposed architecture.

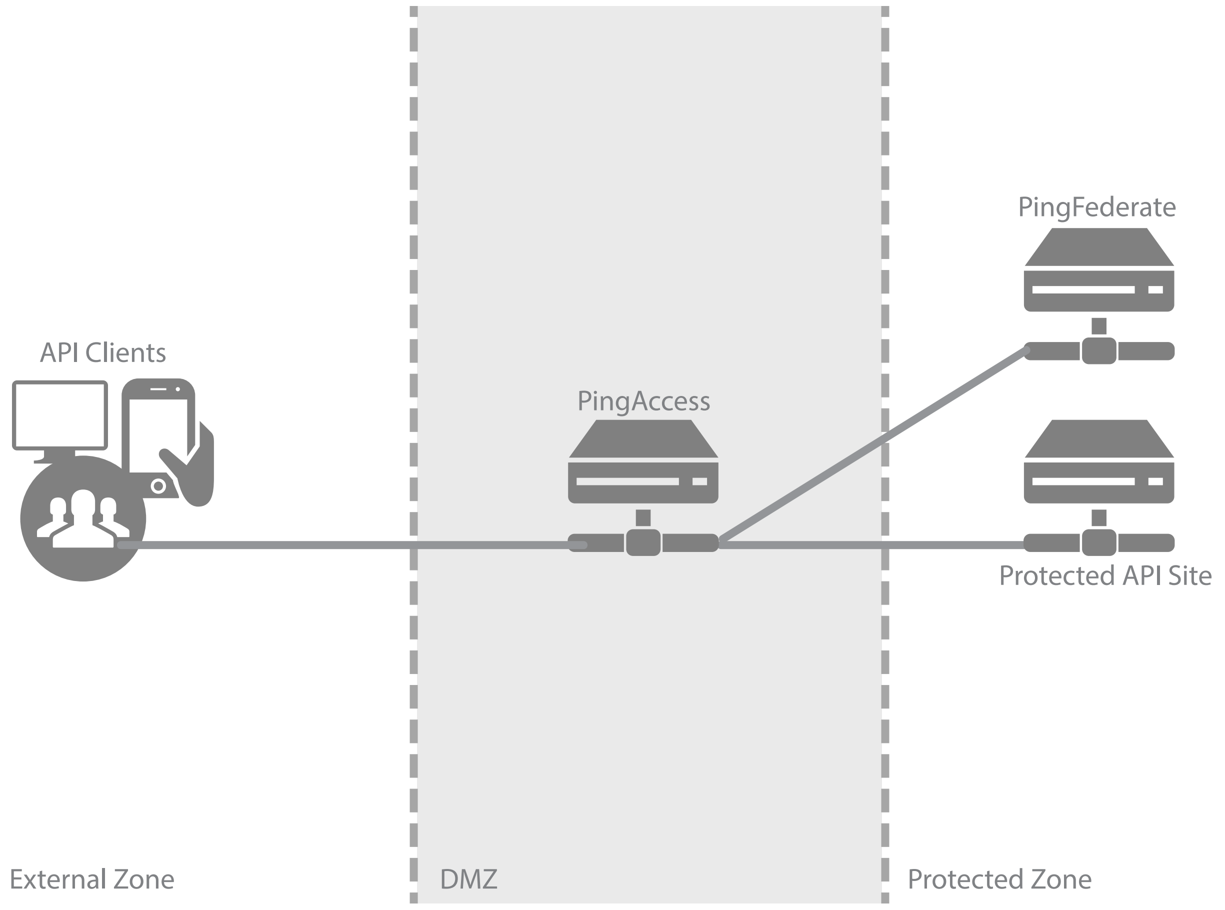
Zone	Description
External Zone	External network where incoming requests for Web applications originate.
DMZ	Externally exposed segment where (possibly multiple) application Web servers are accessible to Web clients. PingAccess Agent is deployed as a plugin on these Web servers. Agents interact with PingAccess Policy Server in the Protected Zone.

Zone	Description
Protected Zone	Back-end controlled zone with no direct access by Web clients. PingAccess Policy Server is deployed in a cluster in this zone with a separate administrative engine. PingFederate is also deployed in this zone in a cluster with its own separate administrative engine. PingFederate needs access to the Identity Management Infrastructure in order to authenticate users. Since during user authentication Web clients need to interact with PingFederate directly, a reverse proxy such as PingAccess Gateway is required to forward client requests through the DMZ. This aspect is not shown in the diagram.

API Access Management proof of concept deployment architecture

This environment is used to emulate an API acccess management environment for testing purposes.

In the test environment, PingAccess can be set up with the minimum hardware requirements. Given these conditions, we do not recommend using this proposed architecture in a production deployment as it does not provide high availability.



The following table describes the three zones within this proposed architecture.

Zone	Description
External Zone	External network where incoming API requests originate.
DMZ	Externally exposing segment where PingAccess is accessible to API clients. PingAccess is a standalone instance in this environment, serving as both a runtime and an administrative port.
Protected Zone	Back-end controlled zone in which Sites hosting the protected APIs are located. All requests to these APIs must be designed to pass through PingAccess. PingFederate is accessible to API clients in this zone and is a standalone instance, serving as both a runtime and an administrative port.

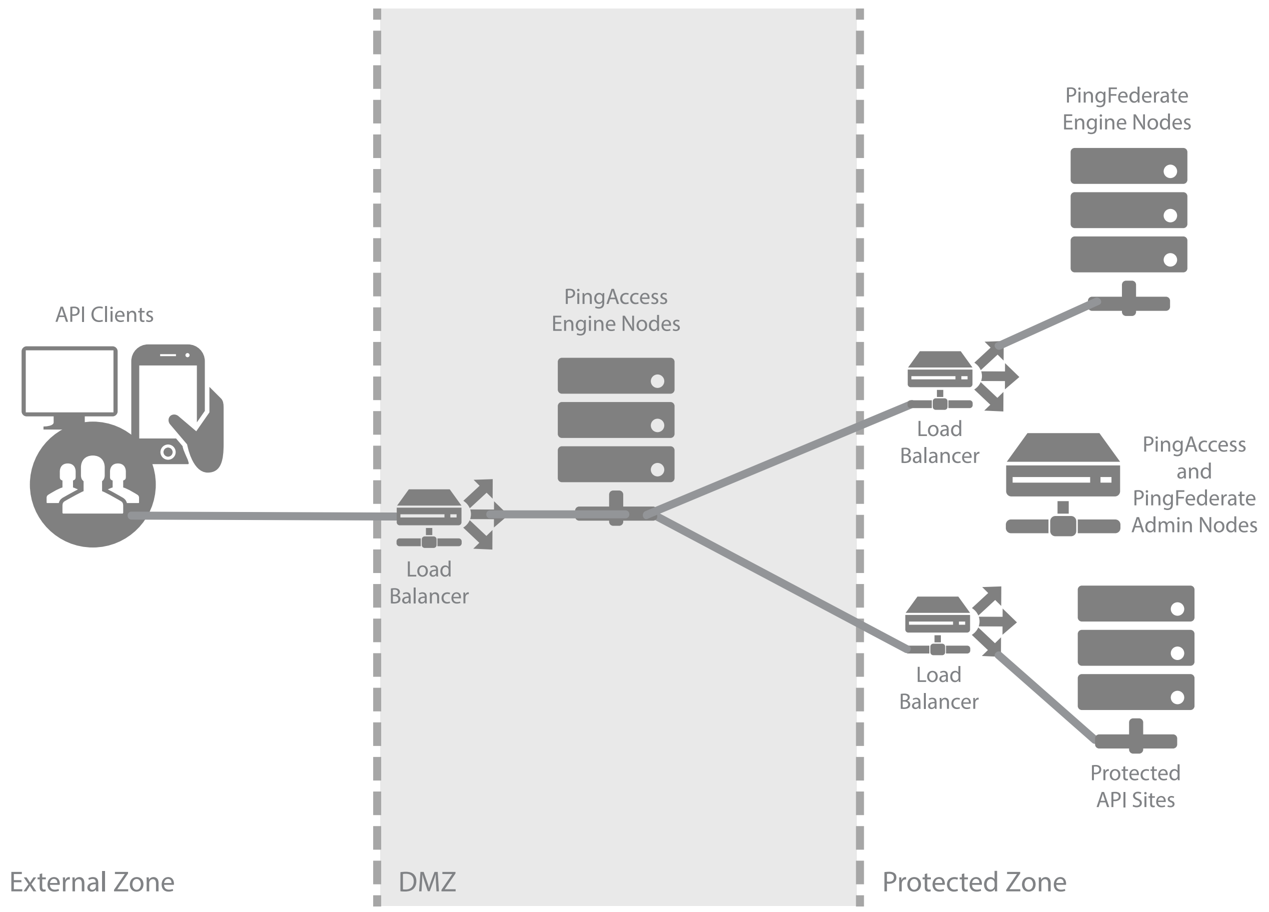
API Access Management production deployment architecture

This environment shows an API acccess management architecture.

There are many considerations when deploying a Production environment. For high availability and redundancy, the environment requires clustering and load-balancing. Load balancers are required as part of the networking infrastructure to achieve high availability by ensuring that requests are sent to available servers they are front-ending. Best practices in network design and security also include firewalls to ensure that only required ports and protocols are permitted across zones.

Info: PingAccess can provide high availability and basic load balancing for the protected web apps in the protected zone. See the Load Balancing Strategies documentation for more information.

The following environment example is a recommended production quality deployment architecture for an API access management use case.



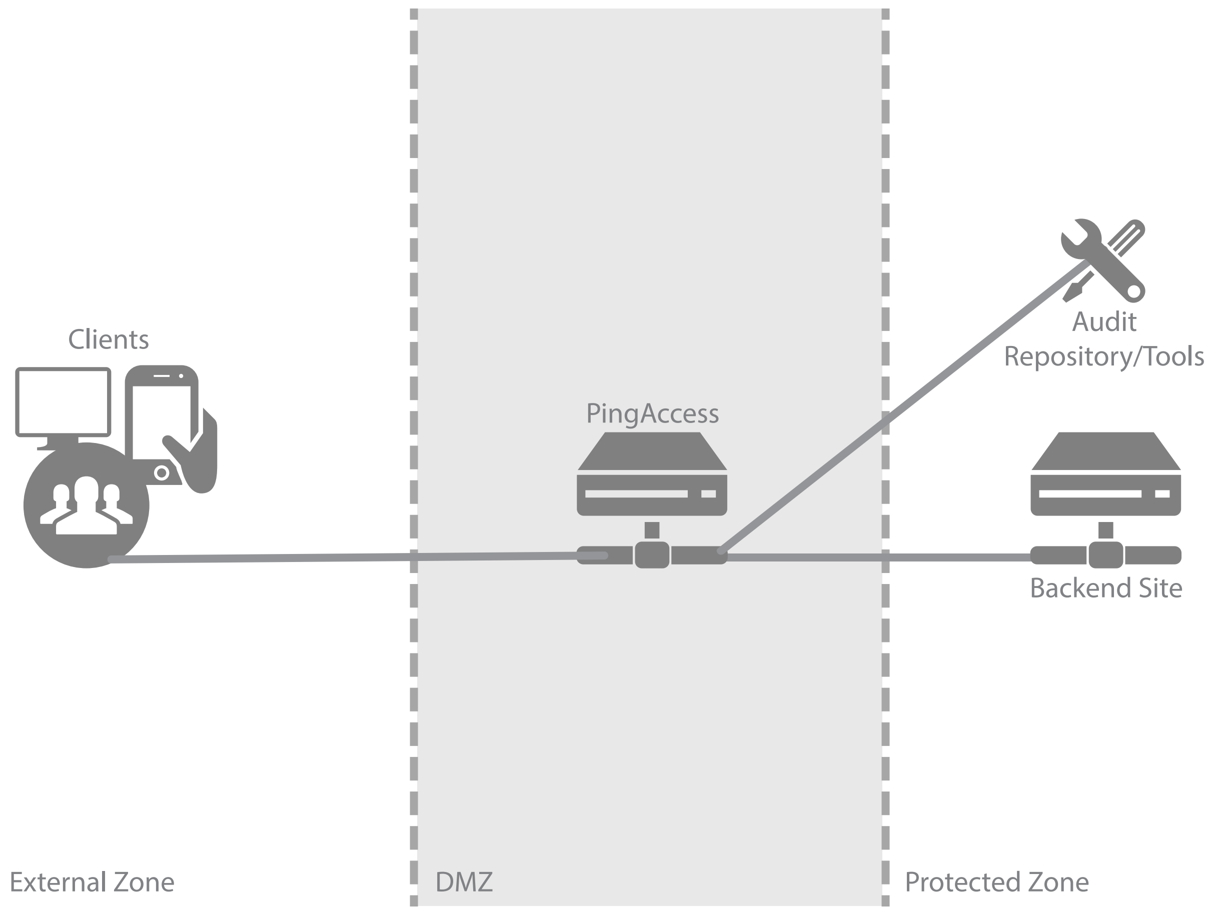
The following table describes the three zones within this proposed architecture.

External Zone	External network where incoming API requests originate.
DMZ	Externally exposing segment where PingAccess is accessible to API clients. A minimum of two PingAccess engine nodes will be deployed in the DMZ to achieve high availability. Depending on your scalability requirements, more nodes may be required.
Protected Zone	Back-end controlled zone in which Sites hosting the protected APIs are located. All requests to these APIs must be designed to pass through PingAccess. PingFederate is accessible to API clients in this zone. A minimum of two PingFederate engine nodes will be deployed in the protected zone. Administrative nodes for both PingAccess and PingFederate may be co-located on a single machine to reduce hardware requirements

Auditing and proxying proof of concept deployment architecture

This environment is used to emulate an auditing and proxying environment for testing purposes.

In the test environment, PingAccess can be set up with the minimum hardware requirements. Given these conditions, we do not recommend using this proposed architecture in a production deployment as it does not provide high availability.



The following table describes the three zones within this proposed architecture.

Zone	**Description**
External Zone	External network where incoming requests originate.
DMZ	Externally exposing segment where PingAccess is accessible to clients. PingFederate and PingAccess are standalone instances in this environment, serving as both runtime and administrative ports.
Protected Zone	Contains back-end Sites audited and proxied through PingAccess. Audit results are sent to an audit repository or digested by reporting tools. Many types of audit repository/tools are supported such as SIEM/GRC, Splunk, database, and flat files.

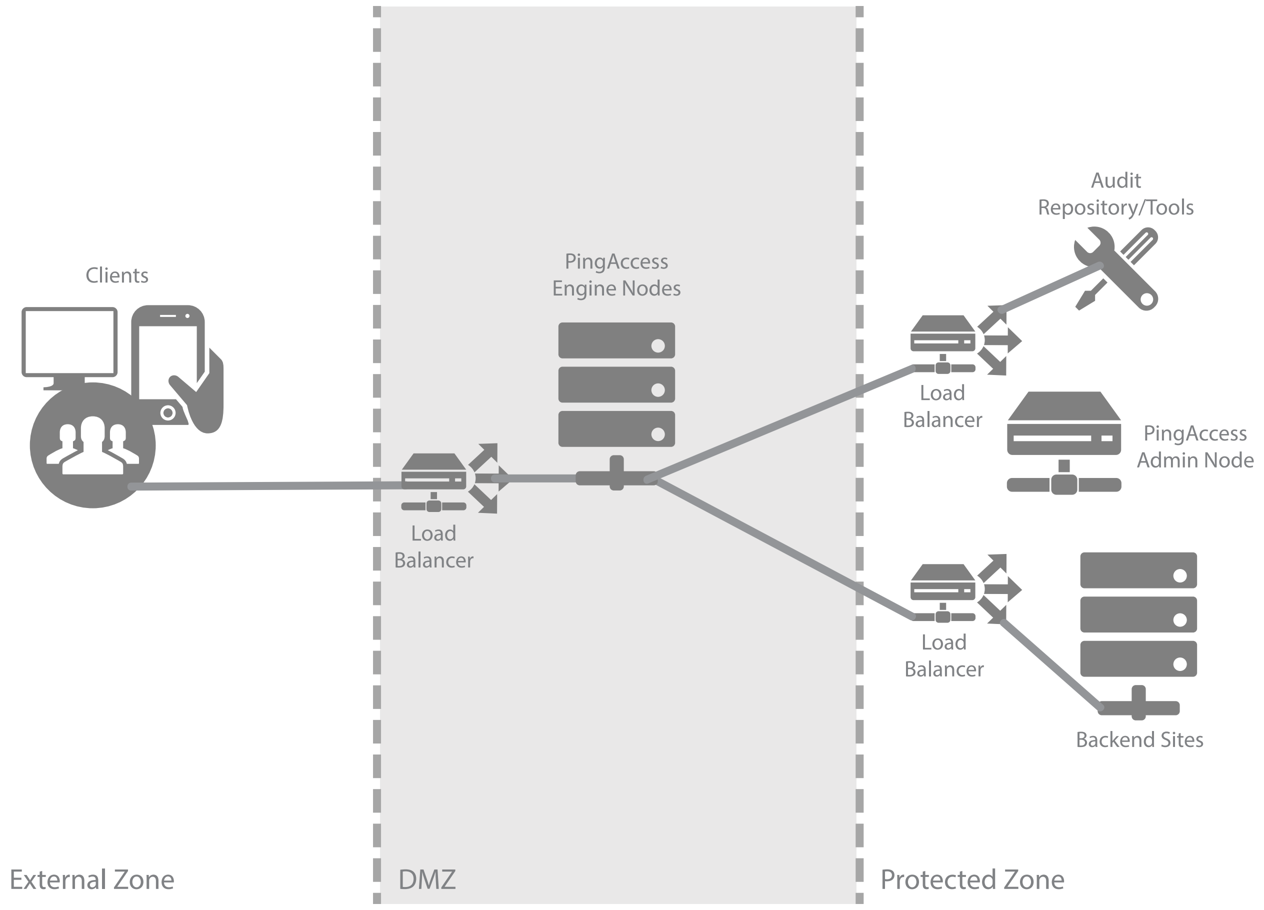
Auditing and proxying production deployment architecture

This environment shows an auditing and proxying architecture.

There are many considerations when deploying a Production environment. For high availability and redundancy, the environment requires clustering and load-balancing. Load balancers are required as part of the networking infrastructure to achieve high availability by ensuring that requests are sent to available servers they are front-ending. Best practices in network design and security also include firewalls to ensure that only required ports and protocols are permitted across zones.

ⓘ **Info:** PingAccess can provide high availability and basic load balancing for the protected web apps in the protected zone. See the Load Balancing Strategies documentation for more information.

The following environment example is a recommended production quality deployment architecture for an auditing/proxying use case.



The following table describes the three zones within this proposed architecture.

External Zone	External network where incoming requests originate.
DMZ	Externally exposing segment where PingAccess is accessible to clients. A minimum of two PingAccess engine nodes will be deployed in the DMZ. Depending on your scalability requirements, more nodes may be required.
Protected Zone	Contains back-end Sites audited and proxied through PingAccess. Audit results are sent to an audit repository or digested by reporting tools. Many types of audit repository tools are supported such as SIEM/GRC, Splunk, database, and flat files.

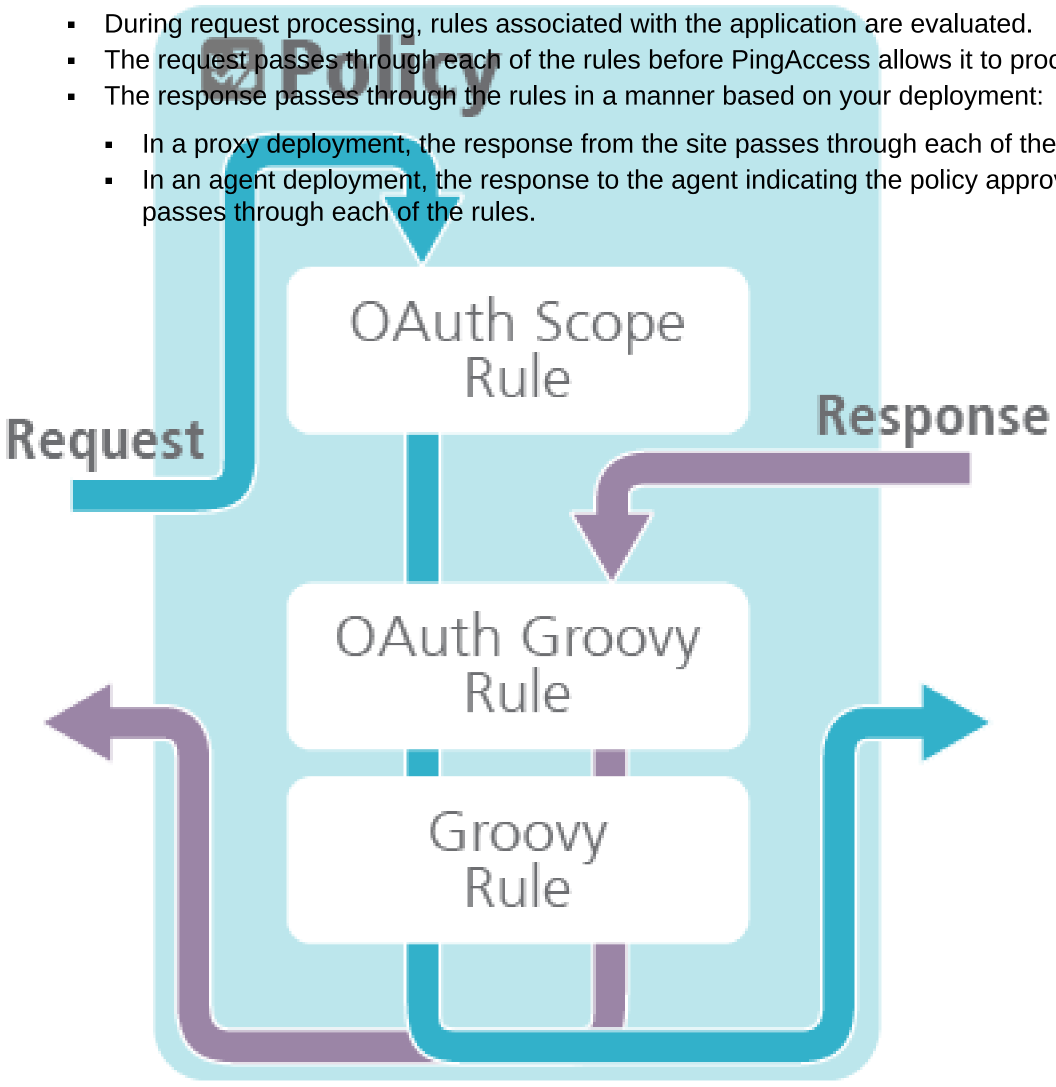
Groovy Development Reference Guide

Groovy

PingAccess provides the Groovy script and OAuth Groovy script rule types that enable the use of Groovy, a dynamic programming language for the Java Virtual Machine (see the *Groovy documentation*). Groovy scripts provide advanced rule logic that extends PingAccess rule development beyond the capabilities of the packaged rules.

Groovy scripts have access to important PingAccess runtime objects such as the *Exchange* and *PolicyContext* objects, which the scripts can interrogate and modify. Groovy script rules are invoked during the request processing phase of an exchange, allowing the script to modify the request before it is sent to the server. Groovy script rules are also invoked during the response, allowing the script to modify the response before it is returned to the client. The diagram below highlights the flow of Rule processing.

- During request processing, rules associated with the application are evaluated.
- The request passes through each of the rules before PingAccess allows it to proceed.
- The response passes through the rules in a manner based on your deployment:
  - In a proxy deployment, the response from the site passes through each of the rules.
  - In an agent deployment, the response to the agent indicating the policy approval or denial passes through each of the rules.



Groovy Scripts

Groovy scripts provide advanced rule logic that extends PingAccess Rule development beyond the capabilities of the packaged rules.

Groovy scripts have access to important PingAccess runtime objects such as the *Exchange* and *PolicyContext* objects, which the scripts can interrogate and modify. Groovy script rules are invoked during the request processing phase of an exchange, allowing the script to modify the request before it is sent to the server. Groovy script rules are also invoked during the response, allowing the script to modify the response before it is returned to the client. See *Groovy* for more info about Groovy.

ⓘ **Note:** Through Groovy scripts, PingAccess administrators can perform sensitive operations that could affect system behavior and security.

Matchers

Groovy scripts must end execution with a Matcher instance. Matchers provide a framework for establishing declarative rule matching objects. You can use a Matcher from the list of *PingAccess Matchers* or from the *Hamcrest library*.

The following are Hamcrest method examples for constructing access control policies with the Web Session Attribute rule using evaluations such as an `OR` group membership evaluation.

allOf

Matches if the examined object matches all of the specified Matchers. In this example, the user needs to be in both the sales and managers groups for this rule to pass.

```
allOf(containsWebSessionAttribute("group","sales"),
 containsWebSessionAttribute("group","managers"))
```

anyOf

Matches any of the specified Matchers. In this example, the rule passes if the user is in any of the specified groups.

```
anyOf(containsWebSessionAttribute("group","sales"),
 containsWebSessionAttribute("group","managers"),
 containsWebSessionAttribute("group","execs"))
```

not

Inverts the logic of a Matcher to not match. In this example, the rule fails if the user is in both the sales and the managers groups.

```
not(allOf(containsWebSessionAttribute("group", "sales"),
 containsWebSessionAttribute("group", "managers")))
```

See *Matchers* for more information.

Objects

The following objects are available in Groovy. Click a link for more information on that object.

Exchange Object

Contains the HTTP request and the HTTP response for the transaction processed by PingAccess.

PolicyContext Object

Contains a map of objects needed to perform policy decisions. The contents of the map vary based on the context of the current user flow.

Request Object

Contains all information related to the HTTP request made to an application.

Response Object

Contains all information related to the site HTTP response.

Method Object

Contains the HTTP method name from the request made to an application.

Header Object

Contains the HTTP header information from the request made to an application or the HTTP header from a Site response.

Body Object

Contains the HTTP body from the application request or the HTTP body from the site response.

OAuthToken Object

Contains the OAuth access token and related identity attributes.

Logger Object

Configure and view the state of logging.

MediaType Object

Contains information related to the media type.

Debugging/troubleshooting

Groovy script rules are evaluated when saved to ensure that they are syntactically valid. If a Groovy script rule fails to save, hover over the information icon to view additional information about the reason for the failure.

If a rule fails when it is run, information about the failure is added to the `<PA_HOME>/log/pingaccess.log` file.

Info: Some error messages about Groovy rule failures are only logged if DEBUG level output is enabled for the `com.pingidentity` logger.

Body object

Purpose

Accesses the Body object in Groovy `exc?.request?.body` or `exc?.response?.body`

The Body object contains the HTTP body from the application request or the HTTP body from the site response. The request HTTP body is sent on to the site after the rules are evaluated. The response HTTP body is sent on to the User-Agent after the response rules are evaluated.

Groovy sample

```
//Checks the actual length of the body content and set the Content-Length
 response header
def body = exc?.response?.body;
def header = exc?.response?.header;
header?.setContentLength(body?.getLength());
pass();
```

Method summary

Method	**Description**
byte[] getContent()	Returns the body content of the request or response.
int getLength()	Returns the length of the body content.

Exchange object

Purpose

Accesses the Exchange object in Groovy - `exc`

The Exchange object is available to both the OAuth Groovy Script Rule and the regular Groovy Script Rule. PingAccess makes the Exchange object available to Groovy Script developers to provide request and response information for custom Groovy Rules.

The Exchange object contains both the HTTP request and the HTTP response for the transaction processed by PingAccess. You can use this object to manipulate the request prior to it being sent to the site. You can also use this object to manipulate the response from the site before it is sent to the client.

An instance of the Exchange object lasts for the lifetime of a single Application request. The Exchange object can be used to store additional information determined by the developer.

Some fields and methods for the Response Object are not available in scripts used with an Agent. See the Field Summary and Method Summary tables below for more information.

Groovy sample

```
//Evaluate if the content length of the request is empty
if (exc?.request?.header?.contentLength > -1 )
{
    //Set a custom header in the request object
    exc?.request?.header?.add("X-PINGACCESS-SAMPLE", "SUCCESS")
    pass()
}
else
{
    println("Request content is empty") //Debugging statement
    fail()
}
```

Method summary

Method	**Description**
Identity getIdentity()	Obtains the PingAccess representation of the identity associated with the request. This object will be null for requests to an unprotected application or an unauthenticated request to an anonymous resource.
Request getRequest()	Obtains the PingAccess representation of the request. This request is sent to the site with any changes that might be made in a Groovy script.

Method	Description
Response getResponse()	Obtains the PingAccess representation of the response. If the site has not been called, the response is null. This field is not available in scripts used with an Agent.
long getTimeReqSent()	Obtains the time, in milliseconds, when the request was sent to the site. This field is not available in scripts used with an Agent.
long getTimeResReceived()	Obtains the time, in milliseconds, when the response was received from the site. This field is not available in scripts used with an Agent.
String getRequestURI()	Returns the PingAccess URI that received the request.
String getRequestScheme()	Obtains the scheme used by the browser or other user agent that made the request.
Object getProperty(String key)	Returns the value of a custom property.
void setProperty(String key, Object value)	Sets a custom property.
SslData getSslData()	Obtains information established in the TLS handshake made with PingAccess.

Headers object

Purpose

Accesses the Headers object in Groovy exc?.request?.header or exc?.response?.header

The Headers object contains the HTTP Header information from the request made to an application or the HTTP Header from a site response. The *Request* HTTP Header is sent on to the site after the Rules are evaluated. The *Response* HTTP Header is returned to the client after the response Rules are evaluated.

Use the Headers object to add custom HTTP headers for site.

Groovy sample

```
if ( !(exc.response) )
{
     // Set a custom header for the Site request
     def header = exc?.request?.header
     header?.add("X-PINGACCESS-SAMPLE", "SUCCESS")
}
pass()
```

Method summary

Method	Description
void add(String key, String val)	Adds HTTP header fields for the request. ⓘ **Info:** Note that if Groovy Rules are used to inject HTTP headers for the backend protected application, the script must sanitize the same headers from the original client request.

Method	**Description**
String getAccept()	Returns the acceptable response Content-Types expected by the User-Agent.
void setAccept(String value)	Sets the acceptable response Content-Types expected by the User-Agent.
String getAuthorization()	Returns the authentication credentials for HTTP Authentication.
void setAuthorization(String username, String password)	Sets authentication credentials for HTTP Authentication.
String getConnection()	Returns the connection type preferred by the User-Agent.
void setConnection(List<String> values)	Sets the connection type preferred by the User-Agent.
int getContentLength()	Returns the request body content length.
void setContentLength(int length)	Sets the request body content length.
MediaType getContentType()	Returns media type of Header with Content type
void setContentType(String)	Sets the request body MIME type.
Map <String, String[]> getCookies()	Returns all cookies sent with the request.
void setCookie(String)	Overwrites the request's cookie header with the passed string. This method cannot be used to set cookies in the response header.
String getFirstCookieValue(String)	Returns the first cookie in the Cookie header.
String getFirstValue(String)	Returns the first value of the HTTP header specified by the name.
void setDate(Date date)	Sets the date of the message in the Date HTTP header.
List<GroovyHeaderFields> getAllHeaderFields()	Returns a list of GroovyHeaderFields.
String getHost()	Returns the hostname specified in the request.
void setHost(String value)	Sets the hostname for the request to the Site.
String getLocation()	Gets the redirect location URL for the response.
void setLocation(String value)	Sets the redirect location URL for the response.
String getProxyAuthorization()	Returns the proxy credentials.
void setProxyAuthorization(String value)	Sets the request proxy credentials.
void setServer(String value)	Sets the server name for the response.
List<String> getValues(String name)	Returns a list of string values for the supplied header name.
String getXForwardedFor()	Returns the originating IP address of the client and the proxies, if set.
void setXForwardedFor(String value)	Sets the IP Address for the client and the proxies.
boolean removeContentEncoding()	Removes the Content-Encoding header value. Returns true if the value has been removed.

Method	Description
boolean removeContentLength()	Removes the Content-Length header value. Returns true if the value has been removed.
boolean removeContentType()	Removes the Content-Type header value. Returns true if the value has been removed.
boolean removeExpect()	Removes the Expect header value. Returns true if the value has been removed.
boolean removeFields(String name)	Removes the header value specified by the name parameter. Returns true if the value has been removed.
boolean removeTransferEncoding()	Removes the Transfer-Encoding header value. Returns true if the value has been removed.

Identity object

Purpose

The Identity object contains information about the authenticated identity associated with the current HTTP request.

Groovy sample

```
// Only allow access for an identity with subject "user"
def subject = exc?.identity?.subject

if ("user".equals(subject)) {
  pass()
} else {
  fail()
}
```

Method summary

Method	Description
String getSubject()	Returns the subject of the identity.
String getMappedSubject()	Returns the subject set by the identity mapping. If there is no identity mapping associated with the application, the return value will be null. If there is an identity mapping associated with the application, but the identity mapping did not determine a subject to map, the returned value may be the empty string.
String getTrackingId()	Returns the tracking identifier used in PingAccess logs. This value is not guaranteed to be globally unique and should be used for diagnostic purposes only
String getTokenId()	Returns the unique ID for the associated authentication token. This value may change when new tokens are issued for the same identity.

Method	Description
Date getTokenExpiration()	Returns a Date object representing the time at which the authentication token expires. This may be null if the authentication provider did not indicate an expiry.
JsonNode getAttributes()	Returns a JsonNode object representing the attributes of the identity.

JsonNode object

Purpose

The JsonNode object represents the attributes of an identity.

Groovy sample

```
// Only allow access if the user is in the group "staff"
def groups = exc?.identity?.attributes?.get("groups")

foundGroup = falseif (groups) {
  for (group in groups) {
    if ("staff".equals(group.asText())) {
      foundGroup = truebreak
    }
  }
}

if (foundGroup) {
  pass()
} else {
  fail()
}
```

Method summary

Method	Description
JsonNode get(String fieldName)	Gets the JsonNode representing a field of this JsonNode. This method will return null if no field exists with the specified name.
boolean has(String fieldName)	Returns true if this JsonNode has a field with the specified name.
java.util.Iterator<String> fieldNames()	Returns an java.util.Iterator providing access to the names of all the fields of this JsonNode.
boolean isTextual()	Returns true if this JsonNode represents a string value.
String asText()	Returns a string representation of this JsonNode. If this JsonNode is an array or object, this will return an empty string.
int intValue()	Returns an integer representation of this JsonNode. If this JsonNode does not represent a number, 0 is returned.

Method	**Description**
boolean isArray()	Returns true if this JsonNode is an array.
boolean isObject()	Returns true if this JsonNode is an object.
int size()	For an array JsonNode, returns the number of elements in the array. For an object JsonNode, returns the number of fields in the object. 0 otherwise.
java.util.Iterator<JsonNode> iterator()	Returns an java.util.Iterator over all JsonNode objects contained in this JsonNode. For an array JsonNode, the returned java.util.Iterator will iterate over all the elements in the array. For an object JsonNode, the returned java.util.Iterator will iterate over all field values in the object.

Remarks

A JsonNode implements java.lang.Iterable<JsonNode> so a for loop can be used to iterate over all the elements in an array JsonNode or the field values in an object JsonNode.

Logger object

Purpose

Accesses the Logger object.

Method summary

Method	**Description**
void trace(String format, Object... arguments)	Logs a TRACE level message based on the specified format and arguments.
void debug(String format, Object... arguments)	Logs a DEBUG level message based on the specified format and arguments.
void info(String format, Object... arguments)	Logs an INFO level message based on the specified format and arguments.
void warn(String format, Object... arguments)	Logs a WARN level message based on the specified format and arguments.
void error(String format, Object... arguments)	Logs an ERROR level message based on the specified format and arguments.
boolean isTraceEnabled()	Checks if the logger instance is enabled for the TRACE level.
boolean isDebugEnabled()	Checks if the logger instance is enabled for the DEBUG level.
boolean isInfoEnabled()	Checks if the logger instance is enabled for the INFO level.
boolean isWarnEnabled()	Checks if the logger instance is enabled for the WARN level.
boolean isErrorEnabled()	Checks if the logger instance is enabled for the ERROR level.

MediaType object

Purpose

Accesses the MediaType object.

Method summary

Method	**Description**
Map getParameters()	Returns a list of parameters.
String getBaseType()	Returns the media base type.
String getSubType()	Returns the media sub type.
String getParameter(String)	Returns a string containing the value of the request parameter.
String getPrimaryType()	Returns the primary media type.

Method object

Purpose

Accesses the Method object in Groovy `exc?.request?.method`

The Method object contains the HTTP Method name from the request made to an application. The HTTP Method is sent on to the Site after the Rules are evaluated.

Groovy sample

```
//Retrieve the HTTP Method name and make different decisions based on the
 method name
def method = exc?.request?.method?.methodName
switch (method) {
     case "GET":
         println("GET")
         break;
     case "POST":
         println("POST")
         break;
     case "PUT":
         println("PUT")
         break;
     case "DELETE":
         println("DELETE")
         break;
default:
     println("DEFAULT")
     pass()
}
```

Method summary

Method	**Description**
String getMethodName()	Returns the name of the HTTP Method (GET, PUT, POST, DELETE, HEAD).

OAuth Token object

Purpose

Accesses the OAuth Token object in Groovy `policyCtx?.context.get("oauth_token")`

The OAuthToken object contains the OAuth access token and related identity attributes. The OAuthToken instance is available only for OAuth Groovy Script Rules.

Groovy sample

```
def scopes = policyCtx?.context.get("oauth_token")?.scopes
def attr = policyCtx?.context.get("oauth_token")?.attributes
def username =
 policyCtx?.context.get("oauth_token")?.attributes?.get("username")?.get(0)
exc?.request?.header?.add("x-scopes", "$scopes")
exc?.request?.header?.add("x-attributes", "$attr")
exc?.request?.header?.add("x-username", "$username")
pass()
```

Method summary

Method	Description
Instant getExpiresAt()	Contains the expiration instant of the OAuth access token.
Instant getRetrievedAt()	Contains the instant that the OAuth access token was retrieved from PingFederate.
String getTokenType()	Contains the type of OAuth access token. (Bearer, JWT).
String getClientId()	Contains the client ID associated with the OAuth access token.
Set getScopes()	Contains the set of scopes associated with the OAuth access token.
Map<String, List<String>> getAttributes()	Contains a map of identity attributes specific to the user.

PolicyContext object

Purpose

Accesses the Policy Context object in Groovy `policyCtx`

The PolicyContext object is a map of objects needed to perform policy decisions. The contents of the map vary based on the context of the current user flow. A common example is OAuth token information stored in an OAuthToken object contained within the context map. In this example, an OAuthToken object is retrieved from the policy context by using the `oauth_token` key. The OAuthToken object is available only for the OAuth Groovy Script Rule.

Groovy sample

```
def oauthToken = policyCtx?.context.get("oauth_token")
```

Method summary

Method	**Description**
Map<String, Object> getContext()	Container for the *OAuthToken object*.
Exchange getExchange()	Returns the exchange a message relates to.

Request object

Purpose

Accesses the Request object in Groovy `exc?.request`

The Request object contains all information related to the HTTP request made to an application. The request instance is sent on to the site after the Rules are evaluated.

Some fields and methods for the Response Object are not available in scripts used with an Agent. See the Field Summary and Method Summary tables below for more information.

Groovy sample

```
//Retrieve the request object from the exchange object
def request = exc?.request
def contentType = request?.headers?.getContentType()
def containsJson = contentType?.matchesBaseType("application/json")
//Check to make sure the request body contains JSON
if (!containsJson) {
fail()
} else {
    pass()
}
```

Field summary

Field	**Description**
String uri	Returns the PingAccess URI that received the request.
void setUri(String)	Sets the PingAccess URI.

Method summary

Method	**Description**
Method getMethod	Contains the HTTP method information from the request sent to the application.
Header getHeader	Contains the HTTP header information from the request sent to the application. ⓘ **Warning: Warning**: Previously executed custom Rules can modify these values.

Method	Description
Body getBody	Contains the HTTP body information from the request sent to the application. This field is not available in scripts used with an Agent. ⓘ **Warning: Warning**: Previously executed custom Rules can modify these values.
Map<String, String[]> getQueryStringParams()	Parses and returns the query string parameters from the request. If the query string parameters cannot be parsed due to formatting errors, this method will throw a URISyntaxException. Groovy scripts that use this method are not required to catch this exception. Scripts that choose not to catch this exception will fail if the query string parameters are invalid.
Map<String, String[]> getPostParams()	Parse the form parameters from the body content of the request, assuming the content is encoded using the encoding defined by the application/x-www-form-urlencoded content type.
void setBodyContent(byte[] content)	Replaces the body content of the request. This method will also adjust the Content-Length header field to align with the length of the specified content.

Response object

Purpose

Accesses the Response object in Groovy `exc?.response`

The Response object contains all information related to the Service HTTP response. The response instance is sent on to the User-Agent after the Rules are evaluated.

The fields and methods for the Response Object are not available in scripts used with an Agent.

Groovy sample

```
if(exc?.response && exc?.identity) {
    exc.response.header.add("PA-Tracking-ID", exc.identity.trackingId)
}
pass()
```

Field summary

Field	Description
int getStatusCode()	Contains the HTTP response status code.
void setStatusCode(int)	Sets the status code from an integer.
String getStatusMessage()	Contains the HTTP response status message.
void setStatusMessage(String)	Sets the status message from a string.

Method summary

Method	Description
boolean isRedirect()	Returns true if the status code is in the 300's.
Header getHeader	Contains the HTTP header information from the response. **Warning:** Previously executed custom Rules can modify these values.
Body getBody	Contains the HTTP body information from the response. **Warning:** Previously executed custom Rules can modify these values.
void setBodyContent(byte[] content)	Replaces the body content of the response. This method will also adjust the Content-Length header field to align with the length of the specified content.

SslData object

Purpose

The SslData object provides access to information established in the TLS handshake with PingAccess.

Groovy sample

```
// Force TLS client authentication
def certChain = exc?.sslData?.clientCertificateChain
if(certChain && !certChain.isEmpty())
{
  pass();
}
else
{
  fail();
}
```

Method summary

Method	Description
List<String> getSniServerNames()	Returns a list of SNI server_names sent by the user agent in the TLS handshake. Empty if the user agent did not utilize the SNI TLS extension.
List<java.security.cert.X509Certificate> getClientCertificateChain()	Returns the certificate chain presented by the user agent in the TLS handshake. Empty if the user agent did not utilize TLS client authentication.

Groovy script examples

OAuth Policy context example

In some instances, it may be necessary to transmit identity information to Sites to provide details of the user attempting to access a Site. In such instances, Groovy scripts can be used to inject identity information into various portions of the HTTP request to the target. In this example, the Site is expecting the identity of the user to be conveyed via the `User` HTTP header. This can be accomplished using the OAuth Groovy Script Rule and the following Groovy script:

```
user=policyCtx?.context.get("oauth_token")?.attributes?.get("user")?.get(0)
exc?.request?.header?.add("User", "$user")
pass()
```

More complex Groovy script logic:

```
test = exc?.request?.header?.getFirstValue("test");
if(test != null && test.equals("foo"))
{
  //rule will fail evaluation if Test header has value 'foo'
  fail()
}
else
{
  //rule will pass evaluation is Test header has value of anything else
  //or isn't present
  pass()
}
```

Set an exchange property named `com.pingidentity.policy.error.info` so the value will be available for the `$info` variable in error templates when an error is encountered. The `$info` variable can be set by a Groovy Script rule or an OAuth Groovy Script rule.

```
exc?.setProperty("com.pingidentity.policy.error.info", "this value will be
 passed to the template in $info variable")
not(anything())
```

Create a white listing rule for certain characters.

```
if (!exc?.request?.uri?.matches("[\\p{Po}\\p{N}\\p{Z}\\p{L}\\p{M}\\p{Zs}\\./
_\\-\\()\\{\\}\\[\\]]*"))
 {
  fail()
 }
 else
 {
  pass()
 }
```

Add a cookie to the response.

```
// Construct the cookie value
value = "cookie-value"
cookieHeaderFieldValue = "ResponseTestCookie=${value}; Path=/"

// Add the cookie on to the response
exc?.response?.header?.add("Set-Cookie", cookieHeaderFieldValue)

pass()
```

Comb an AND and OR, invoking an existing rule Matcher.

```
if ((anyOf(containsWebSessionAttribute("engineering",
 "true"), containsWebSessionAttribute("marketing", "true")) &&
 (containsWebSessionAttribute("manager", "true")))
{pass()
}
else{
fail()
}
```

Matchers

Groovy script rules and OAuth Groovy script rules must end execution with a Matcher instance. This could either be a Matcher from the list of PingAccess Matchers or from the *Hamcrest library* (for more information on Hamcrest, see the *Hamcrest Tutorial*).

Examples

Example 1 - Simple Groovy rule inserts a custom HTTP header

```
test = "let's get Groovy!"
exc?.response?.header?.add("X-Groovy", "$test")
pass()
```

In the sample rule above, the script ends with a call to the Matcher `pass()`. The `pass()` Matcher signals that the rule has passed.

Example 2 - OAuth Groovy rule checks the HTTP method and confirms the OAuth scope

```
//Get the HTTP method name
def methodName = exc?.request?.method?.methodName()
if (methodName == "POST") {
    hasScope("WRITE")
} else {
    fail()
}
```

In the sample rule above, a Matcher is evaluated at the end of each line of execution. The first Matcher used is the `hasScope()` Matcher that confirms if the OAuth access token has the `WRITE` scope. If this is true, the rule passes.

The `fail()` Matcher combination is evaluated when the `methodName` does not equal `POST`. This Matcher combination evaluates to false.

Ping Access Matchers

The following table lists the Matchers available for the Groovy script rule and the OAuth Groovy script rule.

Matcher	Description
`pass()`	Signals that the rule has passed.
`fail()`	Signals that the rule has failed.

Matcher	Description
`inIpRange(String cidr)`	Validates the source IP address of the request against the cidrstring parameter in CIDR notation. When Source IP headers defined in the *HTTP Requests* page are found, the source IP address determined from those headers is used as the source address. For agents, this value is also potentially controlled by the override options on the Agent settings. **Example**: `inIpRange("127.0.0.1/8")`
`inIpRange(java.net.InetAddress ipAddress, int prefixSize)`	Validates the source IP address against the ipAddress and the prefixSize parameters specified individually. When source IP headers defined in the *HTTP Requests* page are found, the source IP address determined from those headers is used as the source address. For agents, this value is also potentially controlled by the override options on the Agent settings. **Example**: `inIpRange(InetAddress.getByName("127.0.0.1"),8)` is equivalent to `inIpRange("127.0.0.1/8")`
`inIpRange(String cidr, String listValueLocation, boolean fallBackToLastHopIp, String... headerNames)`	Validates the source IP address in the first of the specified headerNames using the cidr value. Can be specified as part of a Groovy script as a means of overriding the configuration stored in PingAccess for a specific Groovy script rule. Valid values for the listValueLocation parameter are `FIRST`, `LAST`, and `ANY`. This parameter controls where, in a multivalued list of source IP addresses, the last source should be taken from. If `ANY` is used, if any of the source IP addresses in a matching header match the CIDR value, the Matcher evaluates to `true`. **Example**: `inIpRange("127.0.0.1/8", "LAST", true, "X-Forwarded-For", "Custom-Source-IP")`
`inIpRange(java.net.InetAddress address, int prefixSize, String listValueLocation, boolean fallBackToLastHopIp, String... headerName)`	Validates the source IP address in the first of the specified headerNames using the `address` and `prefixSize` values. In all other respects, this Matcher behaves the same as the version that uses a cidr value for comparison. **Example**: `inIpRange(InetAddress.getByName("127.0.0.1"), 8, "LAST", true, "X-Forwarded-For", "Custom-Source-IP")`

Matcher	Description
`requestXPathMatches(String xPathString, String xPathValue)`	Validates that the value returned by the xPathString parameter is equal to the xPathValue parameter. **Example:** `requestXPathMatches("//header[@name='Host']/text()","localhost:3000")`
`inTimeRange(String startTime, String endTime)`	Validates that the current server time is between the startTime and endTime parameters. **Example:** `inTimeRange("9:00 am","5:00 pm")`
`inTimeRange24(String startTime, String endTime)`	Validates that the current server time is between the specified 24-hour formatted time range between the startTime and endTime parameters. **Example:** `inTimeRange24("09:00","17:00")`
`requestHeaderContains(String field, String value)`	Validates that the HTTP header field value is equal to the value parameter. **Example:** `requestHeaderContains("User-Agent", "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36")`
`requestHeaderContains(Map<String, String> fieldValuesMap, boolean caseSensitive)`	Validates that all of the HTTP header fields map to the associated value. The first fieldValuesMap string contains the HTTP header name, and the second string contains the value to compare the incoming request header value with. The caseSensitive parameter determines whether a case-sensitive comparison is performed on the value. The second string in the fieldValuesMap supports Java regular expressions. If multiple pairs of strings are present in the fieldValuesMap parameter, then all conditions must be met in order for the Matcher to pass. **Example:** `requestHeaderContains(['User-Agent':'Mozilla/5.0', 'Cookie':'JSESSIONID'], false)`

Matcher	Description
`requestPostFormContains(Map<String, String> fieldValuesMap, boolean caseSensitive)`	Validates that all of the HTTP form fields maps to the associated value. The first fieldValuesMap string contains the form header name, and the second string contains the value to compare the incoming request header value with. The caseSensitive parameter determines whether a case-sensitive comparison is performed on the value. **Note:** This Matcher determines whether to use fields passed in the URL or forms with a `content-type` header of `application/x-www-form-urlencoded`. The second string in the fieldValuesMap supports Java regular expressions. If multiple pairs of strings are present in the fieldValuesMap parameter, then all conditions must be met in order for the Matcher to pass. **Example:** `requestPostFormContains(['email':'@example.com', 'phonenumber':'720'], false)`
`requestHeaderDoesntContain(String field, String value)`	Validates that the HTTP header field value is not equal to the value parameter. **Example:** `requestHeaderDoesntContain("User-Agent", "InternetExplorer")`
`requestBodyContains(String value)`	Validates that the HTTP body contains the value parameter. **Example:** `requestBodyContains("production")`
`requestBodyDoesntContain(String value)`	Validates that the HTTP body does not contain the value parameter. **Example:** `requestBodyDoesntContain("test")`
`containsWebSessionAttribute(String attributeName, String attributeValue)`	Validates that the PA token contains the attribute name and value. **Example:** `containsWebSessionAttribute("sub", "sarah")`
`containsACRValues(String value)`	Validates that the PA token contains a matching ACR value.

The following table lists the Matchers available to only the OAuth Groovy rule.

Matcher	Description
`hasScope(String scope)`	Validates that the OAuth access token contains the scope parameter. **Example:** `hasScope("access")`
`hasScopes(String... scopes)`	Validates that the OAuth access token contains the list of scopes. **Example:** `hasScopes("access","portfolio")`
`hasAttribute(String attributeName, String attributeValue)`	Checks for an attribute value within the current OAuth2 policy context. **Example:** `hasAttribute("account","joe")`

Performance Tuning Reference Guide

Performance tuning

While PingAccess has been engineered as a high performance engine, its default configuration may not match your deployment goals nor the hardware you have available.

Consult the following sections to optimize various aspects of a PingAccess deployment for maximum performance.

> **Info:** An additional document related to performance, the PingAccess Capacity Planning Guide, is also available to customers as a performance data reference. This document is available from the *Customer Portal* (ping.force.com/Support).

Java tuning

One of the most important tuning options you can apply to the Java Virtual Machine (JVM) is to configure how much heap (memory for runtime objects) to use.

The JVM grows the heap from a specified minimum to a specified maximum. If you have sufficient memory, best practice is to “fix” the size of the heap by setting minimum and maximum to the same value. This allows the JVM to reserve its entire heap at startup, optimizing organization and eliminating potentially expensive resizing.

By default, PingAccess fixes the Java heap at 512 megabytes (MB). This is a fairly small footprint and not optimal for supporting higher concurrent user loads over extended periods of activity. If you expect your deployment of PingAccess to serve more than 50 concurrent users (per PingAccess node if deploying a cluster), we recommend that you increase the heap size.

Configure JVM crash log in Java startup

You can enable or disable the JV crash log, or change the location where it it is stored.

About this task

The JVM crash log location is specified in `run.bat` (Windows) or `run.sh` (Linux) and is enabled by default.

Steps

1. Open `<PA_HOME>/bin/run.bat` (Windows) or `<PA_HOME>/bin/run.sh` (Linux) for editing.

2. To disable JVM crash log reporting, comment out the line that specifies the JVM crash log location. For example, `#ERROR_FILE="-XX:ErrorFile=$PA_HOME/log/java_error%p.log"`.
3. To enable JVM crash log reporting, remove the comment tag and make the line active. For example, `ERROR_FILE="-XX:ErrorFile=$PA_HOME/log/java_error%p.log"`.

Configure memory dumps in Java startup

You can enable or disable JVM memory dump, or change the location where the dump is stored.

About this task

The JVM memory dump location is specified in `run.bat` (Windows) or `run.sh` (Linux) and is disabled by default.

Steps

1. Open `<PA_HOME>/bin/run.bat` (Windows) or `<PA_HOME>/bin/run.sh` (Linux) for editing.
2. To enable JVM memory dump, remove the comment tag on the line that specifies the JVM memory dump location. For example, `HEAP_DUMP="-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=$PA_HOME/log"`.
3. To disable JVM memory dump, comment out the line. For example, `#HEAP_DUMP="-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=$PA_HOME/log"`.

Modify the Java heap size

You can modify the Java heap size for both Windows and Linux installations, including their services

Steps

1. Open the `jvm-memory.options` file located in `*PA_HOME*/conf`.
2. Specify overall heap size by modifying the `#Minimum heap size` and `#Maximum heap size` parameters.
   - Modify `-Xms512m` to change the `#Minimum heap size` value
   - Modify `-Xmx512m` to change the `#Maximum heap size` value

   Specify units as `m` (megabytes) or `g` (gigabytes).
3. Specify young generation size by modifying the `#Minimum size for the Young Gen space` and `#Maximum size for the Young Gen space` variables.
   - Modify `-XX:NewSize=256m` to change the `#Minimum size for the Young Gen space` value
   - Modify `-XX:MaxNewSize=256m` to change the `#Maximum size for the Young Gen space` value

   Set values to 50% of `#Minimum heap size` and `#Maximum heap size`.

   Info: Not advisable if selecting the G1 collector (see *Garbage Collector Configuration* for more information).
4. If you are running PingAccess as a Windows service, run the `generate-wrapper-jvm-options.bat` file located in `*PA_HOME*/sbin/windows`.

   This file applies the changes from the `jvm-memory.options` file to the `wrapper-jvm-options.conf` file, which is used by the Windows service.

Operating system tuning

This section contains tuning recommendations for your operating system.

The tuning recommendations provided here are particularly useful in preventing deployment issues in high capacity environments. See the included topics for guidance specific to your operating system.

Linux tuning

This section describes tuning recommendations for the Linux operating system environment.

Implement these recommendations to prevent deployment issues, particularly in high capacity environments. The following settings will increase the performance and capacity of the networking (particularly TCP) stack and file descriptor usage, respectively, enabling PingAccess to handle a high volume of concurrent requests.

Network/TCP tuning

Add/Modify the following entries in `/etc/sysctl.conf`:

```
##TCP Tuning##
# Controls the use of TCP syncookies (default is 1)
# and increase the number of outstanding syn requests allowed.
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_max_syn_backlog=8192

# Increase number of incoming connections.
# somaxconn defines the number of request_sock structures allocated
# per each listen call.
# The queue is persistent through the life of the listen socket.
net.core.somaxconn=4096

# Increase number of incoming connections backlog queue.
# Sets the maximum number of packets, queued on the INPUT side,
# when the interface receives packets faster
# than kernel can process them.
net.core.netdev_max_backlog=65536

# increase system IP port limits
net.ipv4.ip_local_port_range=2048 65535

# Turn on window scaling which can enlarge the transfer window:
net.ipv4.tcp_window_scaling=1

# decrease TCP timeout
net.ipv4.tcp_fin_timeout=10

# Allow reuse of sockets in TIME_WAIT state for new connections
# (While this may increase performance, use with caution according
# to the kernel documentation.  This setting should only be enabled
# after the system administrator reviews security considerations.)
net.ipv4.tcp_tw_reuse=1

# Increase the read and write buffer space allocatable
# (minimum size, initial size, and maximum size in bytes)
net.ipv4.tcp_rmem = 4096 65536 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216

# The maximum number of packets which may be queued
# for each unresolved address by other network layers
net.ipv4.neigh.default.unres_qlen=100
net.ipv4.neigh.eth0.unres_qlen=100
net.ipv4.neigh.em1.unres_qlen=100

# Default Socket Receive and Write Buffer
net.core.rmem_default=8388608
net.core.wmem_default=8388608
##############
```

Increase file descriptor limits (systemv)

If PingAccess is run through systemv, add or modify the following lines in `/etc/security/limits.conf`:

```
pingAccessAccount soft nofile value
pingAccessAccount hard nofile value
```

where *`pingAccessAccount`* is the user account used to run the PingAccess java process (or * for all users) and *value* is the new value. A value of 65536 (64K) should be sufficient for most environments.

The number of open file descriptors is limited by the physical memory available to the host. You can determine this limit with the following command:

```
cat /proc/sys/fs/file-max
```

If the file-max value is significantly higher than the 65536 limit, consider increasing the file descriptor limit to between 10% and 15% of the system-wide file descriptor limit. For example, if the file-max value is 810752, you could set the file descriptor limit to 100000. If the file-max value is lower than 65536, the host is likely not sized appropriately for PingAccess.

Increase file descriptor limits (systemd)

If PingAccess is run through systemd, modify the following line under the [Service] section of the `/etc/systemd/system/pingaccess.service` file:

```
LimitNOFILE=value
```

where *value* is the new value. The default value of 65536 (64K) should be sufficient for most environments.

The number of open file descriptors is limited by the physical memory available to the host. You can determine this limit with the following command:

```
cat /proc/sys/fs/file-max
```

If the file-max value is significantly higher than the 65536 limit, consider increasing the file descriptor limit to between 10% and 15% of the system-wide file descriptor limit. For example, if the file-max value is 810752, you could set the file descriptor limit to 100000. If the file-max value is lower than 65536, the host is likely not sized appropriately for PingAccess.

Next, run the following command as root:

```
systemctl daemon-reload
```

Finally, restart the PingAccess service.

Windows tuning
This section describes tuning recommendations for the Windows (version 7 and up) operating system environment.

Implement these recommendations to prevent deployment issues, particularly in high capacity environments. The following settings will increase the performance and capacity of network (specifically the TCP socket) connectivity, enabling PingAccess to handle a high volume of concurrent requests.

Increase the number of available ephemeral ports

1. View ephemeral ports: `netsh int ipv4 show dynamicportrange tcp`
2. Increase ephemeral ports: `netsh int ipv4 set dynamicport tcp start=1025 num=64510`
3. Reboot the machine.
4. View and confirm updated port range: `netsh int ipv4 show dynamicportrange tcp`

Reduce socket TIME_WAIT delay

1. Click **Start# Run**, type `regedit` and click **OK** to open the Registry Editor.
2. Navigate to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\Tcpip\Parameters`.
3. Create a new DWORD value (32 bit) and provide the name `TcpTimedWaitDelay`.
4. Set a decimal value of `30`.
5. Reboot the machine.

Garbage collector configuration

This table provides guidance for configuring the garbage collector.

Selecting the appropriate garbage collector depends on the size of the heap and available CPU resources. The following is a table of available collectors and some general guidance on when and how to use them.

Specify the garbage collector using the `jvm-memory.options` file located in *`PA_HOME`*`/conf`. Modify the parameter beneath `#Use the parallel garbage collector` using the information provided below.

Garbage Collector	Description	Modifications
Parallel	▪ Best used with heaps 4GB or less ▪ Full stop-the-world copying and compacting collector ▪ Uses all available CPUs (by default) for garbage collection	Default collector for server JVM. No modification is required.
Concurrent Mark Sweep (CMS)	▪ Best for heaps larger than 4GB with at least 8 CPU cores ▪ Mostly a concurrent collector ▪ Some stop-the-world phases ▪ Non-Compacting ▪ Can experience expensive, single threaded, full collections due to heap fragmentation	Set to `#XX:+UseConcMarkSweepGC` in `jvm-memory.options`.
Garbage First (G1)	▪ Best for heaps larger than 6GB with at least 8 CPU cores ▪ Combination concurrent and parallel collector with small stop-the-world phases ▪ Long-term replacement for CMS collector (does not suffer heap fragmentation like CMS)	Set to `#XX:+UseG1GC` in `jvm-memory.options`. Also disable `#Minimum size for the Young Gen space` and `#Maximum size for the Young Gen space` tuning. Explicit sizing adversely affects pause time goal. To disable, comment the lines out in the script.

Acceptor threads

You can configure the pool of acceptor threads based on your environment.

PingAccess uses a pool of threads to respond to HTTP/S requests made to the TCP port(s) in use. This applies to both administrative and runtime engine listening ports. Acceptor threads read user requests from the administrative or runtime port and pass the requests to worker threads for processing. For performance, only one acceptor thread need be used in most situations. On larger multiple CPU core machines, more acceptors may be used.

To modify, open the `run.properties` file located in the `conf` directory of your PingAccess deployment and specify the number of acceptors you want to use on the following lines:

`admin.acceptors=N`

`engine.http.acceptors=N`

`agent.http.acceptors=N`

Where *N* represents the number of acceptor threads.

Worker threads

You can modify the minimum and maximum number of worker threads to increase performance.

PingAccess uses a pool of *worker* threads to process user requests and a separate pool to process agent requests. Worker threads receive user requests from Acceptor threads, process them, respond back to the client and then return to the pool for reuse. By default, PingAccess starts with a minimum of five worker threads and grows as needed (unbounded by default). You can define the minimum and maximum number of Worker threads in each pool by adding and/or modifying properties found in the `run.properties` file.

To set values, open the `run.properties` file located in the `conf` directory of your PingAccess deployment. If the properties do not exist in the file add them.

```
engine.httptransport.coreThreadPoolSize=N
engine.httptransport.maxThreadPoolSize=N
```

and

```
agent.httptransport.coreThreadPoolSize=N
agent.httptransport.maxThreadPoolSize=N
```

Where *N* represents the number of worker threads.

Maintenance of the pool is such that if the number of threads in the pool exceeds the value of `engine.httptransport.coreThreadPoolSize`, threads idle for 60 seconds are terminated and removed from the pool. The idle timeout value is not modifiable. However, if the values of `engine.httptransport.coreThreadPoolSize` and `engine.httptransport.maxThreadPoolSize` are the same, a fixed sized pool is created and idle threads are not terminated and removed. Similarly for `agent.httptransport.coreThreadPoolSize` and `agent.httptransport.maxThreadPoolSize`.

Since the pool by default is allowed to grow and shrink based on demand, it is recommended that you tune the `engine.httptransport.coreThreadPoolSize` and `agent.httptransport.coreThreadPoolSize` (minimum) to satisfy moderate demand on the system. We recommend a minimum of 10 threads per available CPU core as a good value to support up to twice the number of concurrent users without error or significant degradation in performance.

Backend server connections

PingAccess provides the Max Connections option to control and optimize connections to the proxied site.

Max Connections

Connections to PingAccess are not explicitly connections to the proxied site. PingAccess creates a pool of connections, unlimited in size by default, that are multiplexed to fulfill client requests. Maintenance of the pool includes creating connections to the site when needed (if none are available) and removing connections when they are closed by the backend server due to inactivity.

In certain situations it can be advantageous to limit the number of connections in the pool for a given Web site. If, for example, the Web site is limited to the number of concurrent connections it can handle or has specific HTTP Keep Alive settings, limiting the number of connections from PingAccess can improve overall performance by not overloading the backend server. In the event that all connections in the pool are in use, a requesting thread waits for one to become available. Assuming that response time from the backend site is sufficiently fast, the time spent waiting for a connection is likely to be less than if the system becomes overloaded.

Info: We strongly recommended that you understand the limits and tuning of the server application being proxied. Setting the **Max Connections** value too low may create a bottleneck to the proxied site, setting the value too high (or unlimited) may cause PingAccess to overload the server.

See Sites documentation for information on setting **Max Connections**.

Logging and Auditing

PingAccess uses a high performance, asynchronous logging framework to provide logging and auditing services with as low impact to overall application performance as possible.

Logging

You can modify your logging settings to increase performance.

Although logging is handled by a high performance, asynchronous logging framework, it is more efficient to the system overall to log the minimum amount of information required. We highly recommend that you review the section of the documentation for logging and adjust the level to the lowest, most appropriate level to suit your needs.

Auditing

You can modify your environment's auditing settings based on your security and performance needs.

As with logging, auditing is provided by the same high performance, asynchronous logging framework. Furthermore, auditing messages can be written to a database instead of flat files, decreasing file I/O. If you do not require auditing for interactions with a Resource or between PingAccess and PingFederate, it is more efficient to disable audit logging. However, if you do require auditing services and have access to a Relational Database Management System (RDBMS), we recommend auditing to a database. You will see a decrease in disk I/O, which may result in increased performance depending on database resources.

Agent tuning

You can modify the properties of your PingAccess agents to improve performance.

Several properties in the `agent.properties` file can be configured for increased performance. See the agent documentation for *Apache* or *IIS* for more information on agent configuration and setting properties.

Max Connections

Connections from the agent to PingAccess are limited by `agent.engine.configuration.maxConnections`. Though the default is set to `10`, the PingAccess policy server sees optimal performance at 50 concurrent requests per CPU. In certain situations it can be advantageous to increase the number of connections. In the event that all connections in the pool are in use, a requesting thread waits for one to become available. Assuming that response time to PingAccess is sufficiently fast, the time spent waiting for a connection is likely to be less than if the system becomes overloaded. Note that this is the maximum number of connections per worker process, and not simply the total number of workers the agent has access to. Setting the `agent.engine.configuration.maxConnections` value too low may create a bottleneck to PingAccess, and setting the value too high may cause PingAccess to become overloaded.

Max Tokens

By default, the maximum number of cached tokens in an agent is unlimited. In certain situations it can be advantageous to limit the size of the cache for the agent, as a smaller cache has a smaller memory footprint, freeing up memory available to the application for servicing requests. However, when the token cache limit is reached, the least recently used token-policy mapping will be removed from the cache. If that token-policy mapping happens to be needed again, the agent will have a cache miss, resulting in the need to obtain a new token-policy mapping from PingAccess.

PingAccess User Interface Reference Guide

PingAccess User Interface Reference Guide

This guide provides a reference for configuration of PingAccess features and components. Use this guide in conjunction with PingAccess use case documentation for a comprehensive set of instructions so you can get the most from your PingAccess implementation.

For ease of use, navigation of this guide is modeled after the PingAccess user interface. To learn more about configuration options for a particular screen, navigate to its topic in the same manner that you navigate the PingAccess user interface.

To learn about PingAccess, including its features and functions, see the *PingAccess Overview*.

Applications

The **Applications** tab contains controls for managing applications, resources, and redirects.

Choose from one of the following sections:

- *Applications* on page 143
- *Global unprotected resources* on page 150
- *Redirects* on page 151

Applications

Applications represent the protected web applications and APIs to which client requests are sent.

Applications are composed of one or more resources, have a common Virtual Host and Context Root, and correspond to a single target site. Applications may also use a common Web Session and Identity Mapping. Access control and request processing rules can be applied to applications and their resources on the Policy Manager page to protect them. Applications can be protected by PingAccess Gateway or PingAccess Agent. In a gateway deployment, the target application is specified as a Site. In an agent deployment, the application destination is an Agent.

There are 3 application types:

- Web
- API
- Web + API

Web + API applications allow administrators to configure both Web and API settings for an application. These applications are able to switch between web and API processing behaviors on the fly based on whether the inbound request contains a web session cookie (web) or an OAuth token (API). If the inbound request contains neither, PingAccess will fallback to the method you specify as the Fallback Type for the application.

Use this page to define the applications which PingAccess protects and to which client requests are ultimately forwarded. You can use resources to partition the application into areas requiring distinct access control. Each Application contains at least a Root Resource. The combination of Virtual Server and Context Root must be unique for each Application.

About SPA support

SPA Support merges the conventional 401 unauthorized response of an API application with the traditional 302 redirect response of a web application when a client request does not contain an authentication token. The SPA Supported result is a 401 response containing a JavaScript body that can initiate a 302 redirect. API clients will ignore the JavaScript body and react appropriately to the 401 response. However, browser clients will disregard the 401 response and execute the JavaScript body resulting in a redirect to the token provider to authenticate. Since clients self-select the portion of the response they are prepared to process, the result is a seamless authentication experience regardless of the client type.

SPA support can be used for all application types. When SPA support is enabled for Web + API applications, where a variety of client types are expected to communicate with the application, a fallback type is no longer required since both web and API clients will be properly redirected to authenticate by the same response. It may also benefit Web or API application types, for example, if a new version of a web application contains JavaScript framework components to call APIs. In this case, SPA Support may help mitigate issues in responding to different client types for authentication but it will not enable the full features of the other application type. You would need to migrate the application to a Web + API configuration in order to take advantage of the full functionality, such as that for authentication, rules, or identity mapping.

Adding an application

You can add a new application.

Steps

1. Navigate to **Applications**# **Applications**.
2. Click **+ Add Application**.
3. Complete the fields on the screen using *Manage Applications - Field Descriptions* as a guide.
4. Click either **Save** or **Save & Go to Resources** when finished. The latter option allows you to configure additional application resources.

ⓘ **Note:** When you save the application, PingAccess verifies the Redirect URI for the Application's virtual host is configured in the PingFederate. If PingAccess determines that the Redirect URI is not defined, you will receive the following warning:

```
Save succeeded. Unable to find a matching Redirect URI in the PingFederate
 OAuth Client configuration for [<VHost>/pa/oidc/cb]
```

If you see this warning, ensure that there is a Redirect URI that matches configured. If you have a wildcard in your Virtual Host configuration, ensure the Redirect URI list includes the same wildcard host definition, otherwise you may have a configuration that is only valid in some circumstances.

This validation is performed if the **Application Type** is `Web` or `Web + API`, a **Web Session** is selected, and the PingFederate Administration connection is configured.

Application Field Descriptions

This table describes the fields available for managing applications at **Applications**# **Applications**.

Field	**Required**	**Description**
Name	**Yes**	A unique name for the application.
Description	No	An optional description for the application.

Context Root	**Yes**	The context at which the application is accessed at the site. **Note:** This value must meet the following criteria: ▪ It must start with `/`. ▪ It can contain additional `/` path separators. ▪ It must not end with `/`. ▪ It must not contain wildcards or regular expression strings. ▪ The combination of the **Virtual Host** and **Context Root** must be unique. The following *is* allowed and incoming requests will match the most specific path first: ▪ `vhost1:443/App` ▪ `vhost1:443/App/Subpath` ▪ `/pa` is, by default, reserved for PingAccess and is not allowed as a **Context Root**. You can change this reserved path using the PingAccess Admin API.
Case Sensitive Path	No	Indicates whether or not to make request URL path matching case sensitive.
Virtual host(s)	**Yes**	Specifies the virtual host for the application. Click **Create** to create a virtual host.
Application Type	**Yes**	Specifies the application type, either `Web`, `API`, or `Web + API`. ▪ If the **Application Type** is `Web`, specify whether or not you want to enable ***SPA Support*** . Select the **Web Session** if the application is protected and, if applicable, the **Identity Mapping** for the application. Click **Create** to create a Web Session or Identity Mapping. ▪ If the **Application Type** is `API`, specify whether or not you want to enable ***SPA Support*** . Indicate the method of **Access Validation** and, if applicable, select the **Identity Mapping** for the application. Click **Create** to create an Identity Mapping. ▪ If the **Application Type** is `Web + API`, specify whether or not you want to enable ***SPA Support*** . Indicate the method of **Access Validation**. Select the **Web Session** and, if applicable, the **Identity Mappings** to use for each type. In this configuration, the Web Session is required and the API is protected by default.
Destination	**Yes**	Specifies the application destination type, either `Site` or `Agent`. ▪ If the destination is a `Site`, select the **Site** requests are sent to when access is granted. If HTTPS is required to access this application, and at least one non-secure HTTP listening port is defined, select the **Require HTTPS** option. Click **Create** to create a Site. ▪ If the destination is an `Agent`, select the agent which intercepts and validates access requests for the Application. Click **Create** to create an Agent.

Enabled	No	Select to enable the application and allow it to process requests.

Editing an application

You can edit an existing application.

Steps

1. Navigate to **Applications**.
2. Expand the application on the **Properties** tab and click ✎.
3. Make the required changes using *Manage Applications - Field Descriptions* as a guide.
4. Click **Save** or **Save and Go to Resources**.

Deleting an application

You can delete an existing application.

Steps

1. Go to **Applications**.
2. Expand the application and click 🗑.
3. Click **Delete** to confirm.

Application Resources

Application resources are components in an application that require a different level of security. You can manage security settings for application resources.

Choose from one of the following topics:

- *Resource ordering*
- *Path patterns*

Resource ordering

Resources have one or more path patterns. When handling requests, PingAccess determines the path pattern that matches and associates the proper resource. When one or more path patterns matches a request, PingAccess uses the first matching pattern it identifies. As such, the order in which path patterns are evaluated is important.

By default, PingAccess orders path patterns automatically so that the most specific patterns are matched first. However, if more explicit control is needed, or if regular expressions are to be used, resource ordering can be enabled to manually specify the order in which path patterns are evaluated.

For example, an application may have three resources, such as:

- `/images/logo.png` (Basic)
- `/images/*` (Basic)
- `/.+/[a-z]\.png` (Regex)

A request to resource `/images/logo.png` is matched by all 3 path patterns, yet each resource may have different policy requirements. Resource ordering allows you to specify which of these path patterns is parsed first, further allowing you to control the policy that is applied to a particular request.

Enable resource ordering

To enable resource ordering, edit an existing application, navigate to the Resources tab, and click ⟳. Modify the resource order as necessary, then click **Save** to persist the changes.

Disable resource ordering

To disable resource ordering, you must first remove any Regex path patterns. Edit an existing application, navigate to the Resources tab, and click . Click **Disable manual ordering**.

Auto-order resources

When resource ordering is enabled, PingAccess can assist in the process by attempting to intelligently order resources based on their path patterns. While editing an application, navigate to the Resources tab, click , then click **Auto Order**. Modify the resource order as necessary, then click **Save** to persist the changes.

> **Important:** The Auto Order function will reorder all resources for an application. You cannot undo this action, though you are able to re-order resources manually as appropriate.

Path patterns

To allow for more flexible resource matching, PingAccess supports two types of path matching patterns, **Basic** and **Regex**.

> **Note:** In order to specify a path pattern as **Basic** or **Regex**, you must enable **Resource Ordering**. When Resource Ordering is not enabled, all path patterns are assumed to be Basic, and are parsed as such.

Basic patterns

Basic path patterns (or “wildcard patterns”) are the default path pattern type. Each pattern defines a path to a specific resource or a pattern that matches multiple paths. Basic patterns may contain any number of “*” wildcards, which match zero to many characters in the path.

For example, the Basic pattern:

```
/path/x/*
```

would match any of the following request paths:

```
/path/x/
/path/x/index.html
/path/x/y/z/index.html
```

Regex patterns

Regex path pattern support is enabled when you enable resource ordering. Regex path patterns allow for more flexibility in resource matching. The supported syntax for regex patterns is that documented by the *RE2 wiki*.

> **Note:** When one or more regex path patterns are defined, resource ordering cannot be disabled. You must delete any regex path pattern entries before you can disable resource ordering.

For example, the Regex pattern:

```
/[^/]+/[a-z]+\.html
```

would match any of the following request paths:

```
/images/gallery.html
/search/index.html
```

but would not match any of these paths:

```
/images/gallery2.html
/search/pages/index.html
/index.html
```

Important: **Use of regex path patterns in agent deployments**

Though **Regex** path patterns will function in an agent deployment, a performance decrease may be incurred because the agent must consult PingAccess for policy decisions on all Regex path pattern resources.

In a deployment with **Basic** path patterns and resource ordering disabled, when a PingAccess agent receives a request for a resource, it consults its policy cache for policy decisions.

Agents are unable to interpret Regex path patterns, so a request to an agent for a resource with a Regex path pattern will result in the agent consulting PingAccess for each policy decision.

In a resource ordering scenario, the agent stops consulting its policy cache if it reaches a Regex path pattern, and continues this behavior for all resources ordered after the Regex path pattern resource, regardless of their type. Thus, the ordering of resources is critical to performance.

For example, consider the following scenario:

```
Application A: context root /, resource ordering enabled
Resource 1, Basic, /content
Resource 2, Regex, /\w+-\w+/.*
Root Resource
```

If **Resource 2** is ordered before **Resource 1**, and a request for **Resource 1** is received by the agent, the agent will not leverage its policy cache, since a Regex path pattern disables caching for the associated resource and all resources after it.

If **Resource 1** is ordered before **Resource 2**, the agent will leverage its policy cache for requests to **Resource 1**.

The agent is only able to consult the policy cache for Basic path pattern resources that are ordered before any Regex path pattern resources. If a Basic path pattern resource is ordered after a Regex path pattern resource, the agent will not consult the policy cache, instead contacting PingAccess directly, and thus a performance decrease may be incurred.

Tip: If you are using Regex path patterns in an agent deployment, and if the order in which resources are ordered is unimportant, Regex path patterns should be ordered at the end of the list. If the order is important, perhaps because a particular policy needs to be applied before others, place the resource where appropriate to ensure the correct policy is being applied at the correct time, while potentially incurring a performance impact.

If your deployment makes extensive use of agents and Regex path patterns, and you are experiencing performance problems, you might consider redeploying these applications in a proxy configuration where possible.

Adding an application resource
You can add an application resource to an existing application.

Steps

1. Navigate to **Applications**.
2. Expand an application and click ✎.

3. Click the **Resources** tab.

> ⓘ **Note:** A group containing all global unprotected resources is displayed on the first resources page. Review this list before adding a resource to ensure that there is no conflict between the new resource's path patterns and any uprotected resource path pattern.

4. To add a resource, click **Add Resource**.

> ⓘ **Info:** To edit a resource, expand the resource and click ✎. To delete the resource, expand the resource and click 🗑.

5. Enter a unique **Name** up to 64 characters, including special characters and spaces.
6. Enter a list of URL path patterns (within the Context Root) that identify this resource. If *resource ordering* is enabled, select the path pattern type, **Basic** or **Regex**.

> ⓘ **Info:** The path pattern must start with a slash (/). It begins after the application context root and extends to the end of the URL.
>
> **a.** If automatic path pattern evaluation ordering is in use (default), patterns may contain one or more wildcard characters (*). No use of wildcards is assumed, thus there is a difference between `/app/` and `/app/*`. If a request matches more than one resource, the most specific match is used.
> **b.** If manual path pattern ordering (resource ordering) is enabled, the use of regular expressions is permitted. When one or more path patterns contain a regular expression, you cannot revert to automatic path pattern ordering unless that path pattern is removed.

> ⓘ **Note:** If resource ordering is enabled, and you have specified a regular expression, ensure you select the **Regex** path pattern type. Failure to do so will cause the pattern to be interpreted incorrectly as a **Basic** text string.

> ⓘ **Note:** The application reserved path cannot be used as a path pattern when the context root is `/`. The default application reserved path is `/pa` (`/pa*`). You can modify the default application reserved path using the PingAccess Admin API.

7. Select the type of **Resource Authentication**:

   - Select **Standard** if the resource requires the same authentication as the root application.
   - Select **Anonymous** if this resource has no authentication requirements. Identity mappings are still applied if the user is already authenticated. Access Control and Processing rules are applied where applicable.
   - Select **Unprotected** if this resource has no authentication requirements. Processing rules are applied where applicable. No application or resource access control policy is applied.

> ⓘ **Note:** These options are not available for unprotected applications. **Web** applications types are unprotected when they do not have an associated web session. **API** applications are unprotected when they are not configured to be protected by an authorization server.

8. If the application type is **API** or **Web + API**, enter the **Methods** supported by the Resource. Leave the asterisk default if the Resource supports all HTTP methods, including custom methods. Defining Methods for a Resource allows more fine-grained access control policies on API Resources. For example, if you have a server optimized for writing data (POST, PUT) and a server optimized for reading data (GET), you may want to segment traffic based on the operation being performed.
9. Select the **Audit** checkbox to log information about the transaction to the audit store.
10. If the application type is **Web + API**, and **SPA Support** is disabled on the root application, indicate whether the application resource should override the fallback type specified for the main application. If

you select **Yes** for this option, select the method to be used for the application resource when a request does not contain a web session cookie or OAuth token.

> **Important:** It is important to carefully consider your configuration when making this selection. Changing the Application Fallback Type can have unexpected effects on Resources that do not override the Fallback. For example, if you configure a Web + API Application with a Fallback Type of Web along with several Resources that do not override the Fallback Type, these Resources will emit a 401 response (rather than a 302 to PingFederate) if you later change the Fallback Type to API on the main application. The PingAccess runtime uses Fallback Type to determine which processing flow (Web or API) to use when the request does not contain a web session or an API OAuth Bearer token. When a request does not contain either of these authentication mechanisms, it will rely on this configuration to determine which processing flow to use.

11. Select the **Enabled** check box to enable the resource.
12. Click **Save**.

Applying rules to applications and resources
You can apply rules, rule sets, and rule set groups to applications and resources. These changes are applied instantly.

Steps

1. Navigate to **Applications**.
2. Expand an application in the list and click ✎.
3. Optional: To manage the policies for a resource, select the **Resources** tab, expand the resource you want to edit, and click ✎.
4. Select the applicable tab. For **Web** applications, select the **Web Policy** tab. For **API** applications, select the **API Policy** tab. For **Web + API** applications, you can configure both **Web Policy** and **API Policy** on separate tabs, as required.
5. Using the radio selection, filter by *Rules*, *Rule Sets*, *Rule Set Groups*, or **Rule Type**.
6. To create a new rule, click **Create Rule**.
7. To apply a rule, rule set, or rule set group, drag a rule from **Available Rules** onto the policy bar.
8. Drag items around in the box to change the order in which they are evaluated at runtime.

> **Note:** Rule ordering can affect PingAccess performance; if an access control rule is more likely to reject access, it should appear near the top of the list to reduce the amount of processing that occurs before that rule is applied. This can be more noticeable if, for example, access control policies are applied along with processing rules. Applying your access control policies first ensures that no processing happens on responses unless the user is determined to be allowed access.

9. Click **-** next to an item to remove it from an application or resource.

Global unprotected resources

Global Unprotected Resources are resources that you specify as unprotected for all applications.

To specify a specific resource as unprotected for a single application, see *Application Resources* on page 146.

Adding a global unprotected resource

You can create a new globally unprotected resource.

About this task

> **Warning:** The following steps describe how to globally unprotect resources. Since any resource captured by the Wildcard Path of any entry is left unprotected **for all applications**, great care must be

taken in planning these entries. To unprotected a resource for a specific application, see *Application Resources* on page 146.

Steps

1. Navigate to **Applications**# **Global Unprotected Resources**.
2. Click **+ Add Global Unprotected Resource**.
3. Enter a **Name** for the entry.
4. Optional: Enter a **Description** for the entry.
5. Optional: Select the **Audit Level** checkbox if you want to record access requests for this resource in the audit store.
6. Specify the **Wildcard Path** that identifies the Global Unprotected Resource. This entry must start with a slash (/) and may contain one or more wildcard characters (*). Examples include:
   - /*.jpg
   - /resources/*.css
   - /*/resources/favicon.ico

   Note: Global Unprotected Resource paths are relative to the application context root. Reserved paths such as `/pa`, `/pa/` or `/pa/*` are allowed at the global level, but will not be evaluated for applications that are configured with a context root of `/`.
7. To enable the Global Unprotected Resource, select the **Enabled** checkbox.
8. Click **Save**.

Editing a global unprotected resource
You can edit an existing global unprotected resource.

Steps

1. Navigate to **Main**# **Applications**# **Global Unprotected Resources**.
2. Expand the Global Unprotected Resource you want to edit and click ✎.
3. Make the required changes.
4. Click **Save**.

Deleting a Global Unprotected Resource
You can delete a global unprotected resource.

Steps

1. Navigate to **Main**# **Applications**# **Global Unprotected Resources**.
2. Expand the Global Unprotected Resource you want to delete and click 🗑.
3. Click **Delete** to confirm.

Redirects
Redirects are used to redirect an incoming request to another target.

To configure a redirect, you map the host and port of an incoming request to that of a different target. At runtime, requests made to the **Source** will be redirected to the configured **Target**. This feature is useful in redirecting HTTP requests to an equivalent HTTPS URL.

Redirects are not associated with applications, but rather with the source:port combination you specify.

Adding a redirect
You can add a new redirect.

Steps

1. Navigate to **Applications# Redirects**.
2. Click **+ Add Redirect**.
3. Specify the **Source** host and port that you want to redirect.
4. Specify the **Target** host and post that indicates the destination for the redirect.
5. Using the **Secure Target** check box, indicate whether or not the target is secure.
6. Specify the HTTP **Response Code** you want to associate with the redirect. `301` is specified as the default.
7. To audit redirects, select the **Audit** check box.

Sites

The **Sites** tab contains controls for sites, site authenticators, and third-party services.

Choose from one of the following sections:

- *Sites* on page 152
- *Site Authenticators* on page 154
- *Third-Party Services* on page 157

Sites

Sites are the target applications, endpoints, or APIs which PingAccess Gateway is protecting and to which authorized client requests are ultimately forwarded to.

Adding a site

You can add a site.

Steps

1. Navigate to **Sites# Sites**.
2. Click **Add Site**.
3. Complete the fields on the screen using *Site field descriptions* as a guide.
4. To configure advanced settings, click **Show Advanced**.
5. Click **Save**.

> ⓘ **Note:** If the target site cannot be contacted, the site is saved and a warning is displayed indicating the reason the site was not reachable.

Editing a site

You can edit the properties of an existing site.

Steps

1. Navigate to **Sites# Sites**.
2. Expand the site you want to edit, then click ✎.
3. Make the required changes.
4. Click **Save**.

> ⓘ **Note:** If the target site cannot be contacted, the site is saved and a warning is displayed indicating the reason the site was not reachable.

Deleting a site

You can delete an existing site.

Steps

1. Navigate to **Sites# Sites**.
2. Expand the site you want to delete.
3. Click 🗑.
4. When prompted, click **Delete** to confirm.

Site field descriptions

This table describes the fields available for managing applications at **Sites# Sites**.

Field	Required	Description
Name	**Yes**	Enter a unique **Site Name** up to 64 characters, including special characters and spaces.
Targets	**Yes**	Specify one or more **Targets**. The format for this is `hostname:port`. For example, `www.example.com:80`.
Secure	**Yes**	Select **Secure** if the Site is expecting HTTPS connections. If the site is configured for **Secure** connections, select a **Trusted Certificate Group** from the list, or select **Trust Any** to trust any certificate presented by the listed targets.
Site Authenticators	No	If the Site requires the use of site authenticators, select one or more authenticators from the list. Click **Create** to create a Site Authenticator. Click **x** to remove a Site Authenticator.
Use Target Host Header	No	Select the check box to have PingAccess modify the `Host` header for the Site's Target Host and Target Port rather than the Virtual Host configured in the application. ⓘ **Note:** When cleared, PingAccess makes no changes to the `Host`header. This is often required by target web servers to ensure they service the HTTP request with the correct internal virtual server definition.
Skip Hostname Verification	No	To skip hostname verification, select the checkbox.
Expected Certificate Hostname	No	If you have not selected to skip hostname verification, enter the name of the host expected in the certificate in the **Expected Certificate Hostname** field. This field is available only if the **Skip Hostname Verification** checkbox is not selected. If left blank, the certificates will be verified using the target hostnames.
Availability Profile	No	Select an *Availability profile*. Click **Create** to create an Availability Profile.
Load Balancing Strategy	No	Select a *Load balancing strategy* if the site contains more than one target. Click **Create** to create a Load Balancing Strategy.

Send Token	No	If your site uses the identity information in the PA Token or OAuth access token, leave this check box selected to include the token in the request to the back-end site. If you do not need the token information, you can clear the check box to remove the PA Token from the request. This excludes unnecessary information and decreases the payload size, which might improve performance.
Maximum Connections	No	Enter the maximum number of HTTP persistent connections you want PingAccess to have open and maintain for the Site. A value of `-1` indicates unlimited connections.
Maximum Websocket Connections	No	If the number of WebSocket connections needs to be limited, enter a value. The default of `-1` indicates no limit.
Use Proxy	No	Select if requests to the site should use a configured proxy. **Note:** If the node is not configured with a proxy, requests are made directly to the site. **CAUTION:** If your proxy uses availability handling to retry multiple targets in the event of a network problem, PingAccess should be configured to use only one target for the site. Unexpected behavior could occur if PingAccess and the proxy are both configured to perform availability handling.

Site Authenticators
Site Authenticators define the authentication mechanism that target sites require to control access.
Adding a site authenticator
You can create a new site authenticator.

Steps

1. Navigate to **Sites# Site Authenticators**.
2. Click **Add Site Authenticator**.
3. Enter a unique **Name**.

Note: Special characters and spaces are allowed. This name appears in the **Site Authenticator** list on the **New Site** page.

4. Select the type of authentication from the list. Choose one of the following authentication types to continue.
   - *Basic authentication site authenticator*
   - *Mutual TLS site authenticator*
   - *Token mediator site authenticator* on page 155

Editing a site authenticator
You can edit the properties of an existing site authenticator.

Steps

1. Navigate to **Sites# Site Authenticators**.
2. Expand the site authenticator you want to edit, then click ✎.

3. Make the required changes.
4. Click **Save**.

Deleting a site authenticator

You can delete an existing site authenticator. A site authenticator cannot be deleted if it is associated with a site.

Steps

1. Navigate to **Sites**# **Site Authenticators**.
2. Expand the site authenticator you want to delete.
3. Click 🗑.
4. When prompted, click **Delete** to confirm.

Basic authentication site authenticator

Use HTTP Basic authentication (username:password) to authenticate a client requesting access to a site that requires basic authentication.

ⓘ **Info:** Obtain the username and password from your target Site provider.

Field	Description
Username	The username required for access to the protected Site.
Password	The password required for access to the protected Site.

Mutual TLS site authenticator

Use Key Pairs to authenticate PingAccess to a target Site. When initiating communication, PingAccess presents the client certificate from a Key Pair to the Site during the mutual TLS transaction. The Site must be able to trust this certificate in order for authentication to succeed.

ⓘ **Tip:** Several setup steps are required for PingAccess certificate management before configuring the Mutual TLS Site Authenticator.

Field	Description
Key Pair	The imported/generated key pair for client authentication. Select the key pair you want to use to authenticate PingAccess to the target Site. To create a key pair, see *Importing an existing key pair* on page 203 or *Generating a new key pair* on page 203.

Token mediator site authenticator

The token mediator site authenticator uses the PingFederate STS to exchange a PA Token for a security token, such as a WAM Token or OpenToken, that is valid at the target site.

ⓘ **Note:** The token mediator site authenticator may benefit from the configuration of a PingAccess *Runtime State Cluster* to provide fault-tolerance for mediated tokens if a cluster node is taken offline.

Field	Description
Token Generator ID	Defines the Instance Name of the Token Generator that you want to use. The Token Generator is configured in **PingFederate** (see PingFederate documentation). If *PingFederate Administration* is configured, and PingFederate has one or more token generators configured, this field becomes a list of available token generator IDs.
Logged In Cookie Name	Defines the cookie name containing the token that the target site is expecting.
Logged Off Cookie Name	Defines the cookie name that the target site responds with in the event of an invalid or expired token. If the PA Token is still valid, PingAccess re-obtains a valid WAM Token and makes the request to the site again. If the site responds with the cookie set as logged off again, PingAccess responds to the client with an access denied page.
Logged Off Cookie Value	Defines the value placed in the Logged Off Cookie to detect an invalid/expired WAM Token event.
Source Token	Defines the token type exchanged for a security token during identity mediation. Select **PA Cookie** for Web access or **OAuth Bearer Token** for API identity mediation.
Send Cookies to Browser	Allows the Token Mediator to send the backend cookie defined in the **Logged In Cookie Name** field back to the browser if the protected application has updated it. If the set-cookie header isn't in the response from the protected site, and the Token Mediator Site Authenticator has a cached token for that session, the Token Mediator Site Authenticator will create a new set-cookie response header based on the **Cookie Domain**, **Cookie Max Age**, **HTTP-Only Cookie** and **Secure Cookie** fields in the UI. The admin now can direct the Token Mediator Site Authenticator to be an active player in returning cookies to the user's browser, even when the protected site isn't doing that. This could be used to enable a seamless SSO experience for users navigating from PingAccess protected applications to those protected by a 3rd party Web Access Management system.
Cookie Domain	Enter the domain of the logged in cookie.
Cookie Max Age	Define the length of time in minutes, that you want the generated logged in cookie to be valid.
HTTP-Only Cookie	Define the logged in cookie as HTTP-Only. An HTTP-Only cookie is not accessible using non-HTTP methods, such as calls via JavaScript (for example, referencing document.cookie).

Field	Description
Secure Cookie	Indicate whether the generated logged in cookie must be sent using only HTTPS connections.
Token Processor ID	Defines the Instance Name of a Token Processor that you want to use. The Token Processor is configured in **PingFederate** (see PingFederate documentation). Specify this value if more than one instance of either the JWT Token Processor or the OAuth Bearer Access Token Processor is defined in PingFederate. If *PingFederate Administration* is configured, and PingFederate has one or more token processors configured, this field becomes a list of available token processor IDs.

Third-Party Services

A third-party service configuration defines the destination for HTTPS outbound calls. These definitions are used by custom plugins to indicate how the HTTP client will communicate with the destination.

The configuration of a third-party is similar to that of a site.

Adding a third-party service

You can add a new third-party service.

Steps

1. Go to **Sites**# **Third-Party Services**.
2. Click **+ Add Third-Party Service**.
3. Complete the fields on the screen using *Third-Party Service Field Descriptions* as a guide.
4. Click **Save**.

Editing a third-party service

You can edit the properties of an existing third-party service.

Steps

1. Go to **Sites**# **Third-Party Services**.
2. Expand the Third-Party Service you want to edit and click ✎.
3. Make the required changes.
4. Click **Save**.

Deleting a third-party service

You can delete an existing third-party service.

Steps

1. Go to **Sites**# **Third-Party Services**.
2. Expand the Third-Party Service you want to delete and click 🗑.
3. Click **Delete** to confirm.

Third-Party Service Field Descriptions

The following table describes the fields available for managing applications at **Sites**# **Third-Party Services**.

Field	Required	Description
Name	**Yes**	Specify a name that identifies the third party service.

Targets	**Yes**	Specify one or more hostname:port pairs used to reach the third party service.
Secure	No	Indicate whether or not the target is expecting a secure connection.
Host Value	No	An optional value used as the Host header field value used in requests to a third party service regardless of the target used.
Skip Hostname Verification	No	For secure connections, select to indicate that the third-party service should not perform hostname verification of the certificate.
Expected Certificate Hostname	No	For secure connections, enter the name of the host expected in the certificate when hostname verification is enabled.
Availability Profile	**Yes**	Indicate the availability profile to use.
Load Balancing Strategy	No	Select the load balancing strategy to use if more than one target is defined.
Maximum Connections	**Yes**	Indicates the maximum number of HTTP persistent connections PingAccess will open and maintain for the service. The default of `-1` indicates unlimited connections.
Use Proxy	No	Indicates that requests to the site should use a configured proxy.

Agents

The **Agents** tab contains controls for managing agents. Agents are web server plugins that are installed on the web server hosting the target application, which intercept client requests to protected applications and allow or deny the request to proceed by consulting the Policy Manager or using cached information.

Agents communicate with the PingAccess Policy Server via the PingAccess Agent Protocol (PAAP) which defines in detail the possible interactions between agents and Policy Server. Agents have a name to identify them and a shared secret to authenticate with to Policy Server. Agents do not need to be unique. There can be any number of agents using the same name and secret and they are all treated equally by Policy Server. This is useful in complex deployments where unique agents would be difficult to manage. Agents can be assigned as the destination for one or more applications by name.

Assigning an agent listener key pair

Before you create an agent, you must import or create an Agent listener key pair and assign it to the AGENT Listener.

Steps

1. Import or Generate a Key Pair. The key pair's subject or subject alternative names list need to include the host or hosts the agent will use to contact the PingAccess Policy Server.
2. Navigate to **Networking**# **Listeners**# **HTTPS Listeners**.
3. In the **AGENT** dropdown, select the Key Pair you want to use, and then click **Save**.
4. Restart PingAccess.

Info: If the environment is clustered, check the `pingaccess.log` file on each engine to ensure replication completed before restarting each engine.

Adding an agent

You can create a new agent.

Steps

1. Navigate to **Agents**# **Agents**.
2. Click **Add Agent**.
3. Complete the fields on the screen using *Agent Field Descriptions* as a guide.
4. To configure advanced settings, click **Show Advanced**.
5. Click **Save & Download** to save the configuration and download `<agent-name>_agent.properties` for use with the PingAccess Agent.

ⓘ **Note:** The **Shared Secret** is generated by PingAccess server and identified on this page with a timestamp. Existing secrets can be deleted by clicking the Remove button in the secret field. If an additional secret is needed, *edit* the agent and click **Save & Download** to generate and download a new Shared Secret.

ⓘ **Note:** PingAccess can generate additional agent `agent.properties` files containing the specified information which can be used to configure the agent plugin. Existing configurations can also be re-downloaded if necessary.

Editing an agent

You can edit an existing agent.

Steps

1. Navigate to **Agents**# **Agents**.
2. Expand an existing agent, then click ✎.
3. Make the required changes.
4. To download the Shared Secret, click the **Download** button. To remove the Shared Secret, click the **Remove** button.
5. Click **Save & Download** to download the file.

Deleting an agent

You can delete an existing agent.

Steps

1. Navigate to **Agents**# **Agents**.
2. Expand an existing agent, then click 🗑.
3. When prompted, click **Delete** to confirm.

Agent Field Descriptions

The following table describes the fields available for managing applications at **Agents**.

Field	Required	Description
Name	**Yes**	Enter a unique alphanumeric **Name** for the agent, up to 64 characters.
Description	No	Enter an optional **Description** for the agent and its purpose.

PingAccess Host	**Yes**	In the **PingAccess Host** fields, enter the **Hostname** and **Port** of the PingAccess server where the agent should send requests. ⓘ **Info:** The PingAccess Hostname and Port may not be the actual host and port that Policy Server is listening to, depending on network routing configuration and network elements such as reverse proxies and load balancers. The PingAccess Host and PingAccess Port are where the agent sends its requests. For example, if you have a cluster of engines behind a load balancer, the PingAccess Host and PingAccess Port values might point to the load balancer rather than directly to an engine host in order to provide fault tolerance for the agent connectivity.
Failover Host	No	In the **Failover Host** fields, enter the **Hostname** and **Port** of the PingAccess server where the agent should send requests in the event of a failover from the PingAccess Host. ⓘ **Tip:** Additional failover hosts may be added via API. For more information, see the *PingAccess API Management Guide*.
Agent Trusted Certificate	**Yes**	Specify the **Agent Trusted Certificate** to export in the agent properties file. The agent uses the selected certificate to communicate with the PingAccess engine via SSL/TLS. PingAccess gathers these certificates from imported certificates. If the appropriate certificate is not available, it needs to be *imported into the system*. ⓘ **Note:** You can specify the CA root certificate if the agent listener presents a CA-signed certificate chain.
Override Request IP Source Configuration	No	If required, select **Yes** to **Override Request IP Source Configuration** and enable additional controls that configure the agent to use different IP Source information. ▪ Enter the **Header Names** used to identify the source IP address. ▪ If more than one value is included in the **Header Names** field, use **List Value Location** to specify whether the first value or the last value in the list is used as the source address. The default value is **Last**. ▪ Select **Fall Back to Last Hop IP** to use the last hop IP address as the source address when none of the listed **Header Names** are found. When this option is not selected, if none of the listed **Header Names** are found, access is denied and a `Forbidden` result is returned.

Override Unknown Resource Configuration	No	If required, select **Yes** to **Override Unknown Resource Configuration** to specify how requests for unknown resources should be handled. This mode is optional. If not set, the default agent mode will be used. Select a **Mode** to specify how requests for unknown resources should be handled, either **Deny** or **Pass-Through**.
Max Retries	**Yes**	Enter the **Max Retries** before considering a PingAccess server unavailable.
Failed Retry Timeout	**Yes**	Enter, in seconds, the **Failed Retry Timeout** before retrying a failed PingAccess server.

Rules

The **Rules** tab contains controls for adding and managing rules. Rules let you specify who can access your applications and resources, how and when they can do so, and what modifications should be made to the requested content.

The Policy Manager is a rich drag-and-drop interface where you can manage policies by creating Rules, building Rule Sets and Rule Set Groups, and applying them to Applications and Resources. Policies are rules, set of rules, or groups of rule sets applied to an application and its resources. Policies define how and when a client can access target Sites. When a client attempts to access an application resource identified in one of the policy's Rules, Rule Sets, or Rule Set Groups, PingAccess uses the information contained in the policy to decide whether the client can access the application resource and whether any additional actions need to take place prior to granting access.

Access control rules can restrict access in a number of ways such as testing user attributes, time of day, request IP addresses, or OAuth access token scopes.

Tip: Ensure that any headers used in access control rules (such as `X-Forwarded-For`, which is used by Network Range rules) are sanitized and managed exclusively by inline infrastructure that users must be routed through before reaching PingAccess and the protected applications.

Processing rules can perform request processing such as modifying headers or rewriting URLs.

Info: Processing rules cannot be used with Agents.

Access control rules are applied before processing rules. For each type of rule, the rules are applied in the order configured in the user interface. All rules are evaluated after identity mappings, so rules have access to the request header field set by the identity mapping.

If rules for an application and rules for a resource both apply to a request, this order is used:

1. Application access control rules
2. Resource access control rules
3. Resource processing rules
4. Application processing rules

Manage Rules

You can manage the rules that control access to your web apps and APIs.

Configure advanced fields for rules

You can customize the action to take if policy evaluation fails. The default action is to use a *Rejection Handler* that defines whether to display an error template or redirect to a URL. To use a rejection handler, select it from the list.

With **Basic** rejection handling, you can customize an error message to display as part of the default error page rendered in the end-user's browser if Rule evaluation fails. This page is among the templates you can modify with your own branding or other information. Use the following fields, available in the **Advanced - Rejection Handling** section of a rule, to configure basic error handling.

Field	Description
Error Response Code	The HTTP status response code you want to send if Rule evaluation fails. For example, `403`.
Error Response Status Message	The HTTP status response message you want to return if Rule evaluation fails. For example, `Forbidden`.
Error Response Template File	The HTML template page for customizing the error message that displays if Rule evaluation fails. This template file is located in the *`PA_HOME`*`/conf/template/` directory.
Error Response Content Type	The type of content for the error response so the client can properly display the response.

Rule Creation
PingAccess supports a variety of rule types, and the procedures for rule creation are different for each rule type.

Adding an authentication requirements rule
The Authentication Requirements Rule is a PingAccess access control rule used to limit access to resources or applications protected by PingAccess based on the ACR values returned by the PingFederate Requested AuthN Context Authentication Selector. This allows authentication requirements to be applied when a policy decision is being made by the PingAccess Engine, allowing an entire application or individual resources to require a particular authentication type.

Before you begin

Prerequisites:

- A PingFederate configuration that uses the `Requested AuthN Context Authentication Selector`.
- A configured authentication list.

About this task

Use of this rule also allows for configurations that require more secure authentication methods. For example, a web site might allow a user to authenticate and view personal data using only a username and password, but editing their personal data could require an additional PingID verification step. When used in this manner, an additional step-up authentication event is automatically triggered.

Tip: When used in a rule set with `Any` criteria, this rule should be positioned first in the list to ensure step-up authentication is triggered upon rule set criteria failure.

To configure an Authentication Requirements rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.

3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Authentication Requirements**.
5. Select an *Authentication Requirements List*.
6. Select a **Minimum Authentication Requirement**.

> **Note:** The possible values for the **Minimum Authentication Requirement** are derived from the selected Authentication Requirements list.

7. Click **Save**.

Adding a Cross-origin Request rule
You can add a Cross-origin request rule, which uses cross-origin resource sharing (CORS) to let a web server grant access to restricted resources (fonts, JavaScript, images, etc.) to an application served by another domain without granting access to those resources beyond a list of predefined origin servers.

About this task

Before a CORS request is sent, the originating web server generally sends a "pre-flight" `OPTIONS` request if the request from the client includes credentials. This pre-flight request is used to determine if the target server permits CORS requests to be processed from the originating web server.

PingAccess can be used to evaluate the headers provided in a CORS request to grant or deny access to resources.

> **Note:** In addition to allowing PingAccess to evaluate the CORS request, you can also allow the request to be handled by the protected application, and let PingAccess be excluded from the process of evaluating the access request, if the target application type is `API`. In order to do this with a resource path that is protected by PingAccess and requires user authentication, configure a second resource with the same path pattern, but set the **Methods** field to `OPTIONS` and the **Anonymous** option needs to be cleared. This configuration allows the API request being made to be handled anonymously.

To Configure a Cross-Origin Request Rule

Steps

1. Navigate to **Rules#** **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Cross-Origin Request**.
5. Enter one or more **Allowed Origins** values, clicking ➕ to add additional values.

> **Important:** While it is allowed, we recommend against using a value of `*` in this field. While this is a valid configuration, it is considered to be an insecure practice.

6. If additional options need to be configured, click **Show Advanced**.
7. To permit user credentials to be used in determining access, enable **Allow Credentials**.
8. To modify the **Allowed Request Headers** values, use the following options:

   - To add a new header, click **New Value**.
   - To edit an existing header, click the field and make your changes.
   - To remove an existing header, click 🗑.

   The default headers are `Authorization`, `Content-Type`, and `Accept`.

9. To make specific response headers available to the client that originated the cross-origin request, enter the headers in the **Exposed Response Headers** field. Click **New Value** to add additional headers to the list.
10. To define the request methods allowed in cross-origin requests, select the desired overrides in the **Overridden Request Methods** field.
11. To modify the amount of time the pre-flight **OPTIONS** request is cached, enter the maximum age (in seconds) desired in the **OPTIONS Cache Max Age** field. The default is 600 seconds.
12. Click **Save**.

Rewrite rules overview
It is sometimes necessary to manipulate requests to sites and their responses. PingAccess allows for the manipulation of the Request URI, the cookie domain, the cookie path, and three of the response headers (`Location`, `Content-Location`, and `URI`), as well as the response content.

For example, a site is hosted on `https://server1.internalsite.com` under `/content/`. Users access the site via the following URL in their browser:

`https://server1.internalsite.com/content/`

For example purposes, let's say this results in a *302 Redirect* to an `importantContent.html` page as well as setting a domain cookie for `.internalsite.com`. If you protect this site with PingAccess (using the virtual host `publicsite.com`) under the application `/importantstuff/`, you need to rewrite the content. The information below discusses an example scenario.

Info: This example assumes that a virtual host, a site, and an application are already configured.

Create a Rewrite Content rule

A Rewrite Content Rule alters content in the HTTP Response body.

- In the Response Content-Types field, you define a response type of `text/html`.
- In the Find and Replace criteria, you specify `<a href="https://server1.internalsite.com/content/">` and `<a href="https://publicsite.com/importantstuff/">`.
- Add the Rule to the application. A query to a page with links in it that point to `https://server1.internalsite.com/content/` now point to `https://publicsite.com/importantstuff/`.

Create a Rewrite Cookie Domain rule

A Rewrite Cookie Domain Rule allows the rewriting of the Domain field on cookies when they are set by the back-end site.

- In the Server-Facing Cookie Domain, you enter `internalsite.com`.
- In the Public-Facing Cookie Domain, you enter `publicsite.com`.
- Add the Rule to the application.

  Cookies associated with the domain `publicsite.com` (or `.publicsite.com`) are rewritten to pertain to `internalsite.com` (or `.internalsite.com`).

Create a Rewrite Cookie Path rule

A Rewrite Cookie Path Rule converts the cookie path returned by the Site into a public-facing path.

- In the Server-Facing Cookie Path field, you enter `/content/`.
- In the Public-Facing Cookie Path field, you enter `/importantstuff/`.
- Add the Rule to the application.

  Cookies associated with a cookie path of `/content/` are rewritten to pertain to `/importantstuff/`. After configuring the rewrite Rules as discussed above, a user could access the `https://`

`publicsite.com/importantstuff/` and PingAccess would route that request to `https://server1.internalsite.com/content/`. If the Site sends a redirect to `https://server1.internalsite.com/content/index.html`, PingAccess would return a redirect to `https://publicsite.com/importantstuff/index.html`. If the Site then returned a cookie with a domain of `.internalsite.com` and a path of `/content/`, PingAccess would rewrite that cookie to be relevant to `.publicsite.com` and `/importantstuff/`.

Create a Rewrite Response Header rule

A Rewrite Response Header Rule alters the response header used in the 302 Redirect.

- In the Server-Facing URI field, you enter `https://server1.internalsite.com/content/`.
- In the Public Path field, you enter `/importantstuff/`.
- Add the Rule to the application. A query resulting in a response containing a 302 Redirect to `https://server1.internalsite.com/content/` is rewritten to `https://publicsite.com/importantstuff/`.

> **Info:** This also works for relative redirects: `/content/` is rewritten to `/importantstuff/`. It also works for the path beneath the one defined in the URI: `/content/news/index.html` is rewritten to `importantstuff/news/index.htm`.

Create a Rewrite URL rule

A Rewrite URL Rule alters the Request URI.

- In the Map From field, you enter `^/importantstuff/(.*)` as the regex of the URL's path and query you want to match.
- In the Map To field, you enter `/content/$1`.
- Add the Rule to the application. A query to `https://publicsite.com/importantstuff/` results in PingAccess routing that query to `https://server1.internalsite.com/content/`.

Adding a Rewrite Content rule

You can add a Rewrite Content rule, which modifies text in HTTP response bodies as it is served to the client.

About this task

This rule uses a subset of the Java Regular Expression syntax that excludes look-behind constructs (for example, `\b`) and the boundary matcher (`\G`). If no Java regular expression syntax is used, the effect is to perform a case-sensitive search and replace. The most common use case for this rule is to rewrite host names within URLs contained in HTML, JavaScript or CSS content.

> **Important:** While adding a Content Rewrite Rule in PingAccess 4.2, you cannot add a new row via the UI. Clicking the link to add a new row may cause page controls to become unresponsive. To workaround this issue, use the API to create the rule or create multiple Content Rewrite Rules in the UI and combine them in a rule set.

> **Info:** Extensive use of Rewrite Content Rules may have significant performance implications.

This rule supports content that is either chunked or streamed from the target server. When sent to the client, the content is always chunked.

To configure a Content Rewrite rule:

Steps

1. Navigate to **Rules**# **Rules**.

2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rewrite Content**.
5. Enter one or more **Response Content-Types** to define what type of response data the rewrite rule applies to. The default values are `text/html`, `text/plain`, and `application/json`. The list is an ordered list.

> ⓘ **Info:** Only text-based content types are supported. Text-based content types compressed with gzip, deflate, or compress will be decompressed prior to rewrite rule processing, however the content is not then re-compressed before being sent to the client.

6. Define one or more set of **Find and Replace Criteria**. If multiple criteria are specified, each operation is performed against the original content - effectively applying the rule concurrently.

> ⓘ **Important:** Changes can affect CSS, Javascript, and other text-based elements served to the client. Be sure to properly craft the regular expression to avoid modifying content that wasn't intended.

7. If necessary, increase the size of the buffer used to perform the replace operation by clicking **Show Advanced** and entering a value in **Maximum Buffer Size**.

> ⓘ **Note:** Replacement values cannot be larger than the buffer size. The minimum buffer size that can be specified is 1024 bytes; there is no maximum value.

8. If the protected application does not return a `Content-Type` header, select **Missing Content-Type Allowed**.
9. If **Missing Content-Type Allowed** is enabled, you must specify the encoding the application returns in the **Missing Content-Type Charset** field. For example, this field could contain `UTF-8`. A list of valid values is available in this *Oracle Java 8 SE Technical Note*.
10. Click **Save**.

Example

Examples:

Example description	Original content	Content-type	Find criteria	Replacement value	Modified text
Rewrite URL portion of a web link	<a href="https://serverx.inside.corp/app/">	`text/html`	`serverx.inside.corp`	`www.acme.com`	<a href="https://www.acme.com/app/">
Case-sensitive text replacement	ACMEcorp	`text/html`	Ecorp	`E Corporation`	ACME Corporation
JSON Value masking	`{ "origin": "127.0.0.1, 192.168.1.1" }`	`application/json`	`(127.0.0.1, ).*"`	`***********"`	`{ "origin": "127.0.0.1, ***********" }`

Example description	Original content	Content-type	Find criteria	Replacement value	Modified text
Replacing text inside a specified element using Java regex groups	This text is **bold**.	`text/html`	`<b>(bold)</b>`	`not $1`	This text is not bold.
Case-insensitive text replacement using a Java regex match flag	HTTP	`text/html`	`(?i)http`	`FTP`	FTP

Adding a Rewrite Cookie Domain rule

You can add a Rewrite Cookie Domain rule, which converts the cookie domain returned by the site into a public-facing domain.

About this task

For example, a Site places a cookie on a cookie domain such as `internalsite.com` (or `.internalsite.com`). Using the information configured in the Rewrite Cookie Domain Rule, PingAccess rewrites the `Domain` portion of the `Set-Cookie` response header with a public-facing domain such as `publicsite.com` (or `.publicsite.com`).

ⓘ **Info:** You should only set the cookie (in the Public-Facing Cookie Domain field) to the virtual host name associated with that application or to a domain that is above. For example, `myserver.acme.com` can be set to `acme.com`.

To configure a Rewrite Cookie Domain rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rewrite Cookie Domain**.
5. If the target host needs to be explicitly defined, clear the **Any Site Target Host** checkbox.

   When the **Any Site Target Host** checkbox is enabled, PingAccess will rewrite the cookie domain if it is set to the domain defined in a site's target host list.
6. If **Any Site Target Host** is cleared, enter the domain name to used by the back-end site in the **Server-Facing Cookie Domain** field.
7. In the **Public-Facing Cookie Domain** field, enter the domain name you want to display in the response from PingAccess.
8. Click **Save**.

Adding a Rewrite Cookie Path rule

You can add a Rewrite Cookie Path rule, which converts the cookie path returned by the Site into a public-facing path.

About this task

This enables the details of exposed applications to be managed by PingAccess for security and request routing purposes. For example, a Site places a cookie in a server-facing cookie path such as `/content/`. Using the information configured in the Rewrite Cookie Path Rule, PingAccess rewrites the `Path` portion of the `Set-Cookie` response header with a public-facing cookie path such as `/importantstuff/`.

To configure a Rewrite Cookie Path rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rewrite Cookie Path**.
5. In the **Server-Facing Cookie Path** field, enter the path name where the cookie is valid for the back-end Site.
6. In the **Public-Facing Cookie Path** field, enter the path name you want to display in the response from PingAccess.
7. Click **Save**.

Results

Adding a Rewrite Response Header rule

You can add a Rewrite Response Header rule, which converts the response header value returned by the Site into a public-facing value.

About this task

This Rule rewrites one of three response headers: `Location`, `Content-Location`, and `URI`. For example, the server-facing `Location` response header includes a path that begins with `/test-war/`. Using the information configured in the Rewrite Response Header Rule, PingAccess rewrites `http://private/test-war/` with a public-facing path such as `http://public/path/`.

To configure a Rewrite Response Header rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rewrite Response Header**.
5. If the target host needs to be explicitly defined, clear the **Any Site Target Host** checkbox.

   When the **Any Site Target Host** checkbox is enabled, PingAccess will rewrite the response header URI if it contains a domain defined in a site's target host list.
6. If **Any Site Target Host** is cleared, enter the domain name to used by the back-end site in the **Server-Facing URI** field.
7. In the **Public Path** box, enter a valid URI path that you want to write into the URI. This must be a valid URI path and begin and end with a slash (`/`). For example: `/importantstuff/` or `/`
8. Click **Save**.

Adding a Rewrite URL rule

You can add a Rewrite URL rule, which examines the URL of every request and determines if a pattern matches.

About this task

For example, you define a regular expression (regex) in the rule. If a pattern matches, PingAccess uses the information configured in the Rewrite URL Rule and rewrites that portion of the URL into a path that the Site can understand. The following table displays four example Rewrite URL Rule configurations:

Map from value	Map to value	Example request	Rewrite by PingAccess
`/bank/`	`/application/`	`/bank/content.html`	`/application/content.html`
`^/bank/(.*)`	`/application/$1`	`/bank/content.html`	`/application/content.html`
`/bank/index.html`	`/application/index.jsp`	`/bank/index.html`	`/application/index.jsp`
`/bank/index.html`	`/application/index.jsp`	`/bank/index.html?query=stuff`	`/application/index.jsp?query=stuff`

To configure a Rewrite URL rule:

Steps

1. Navigate to **Rules# Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rewrite URL**.
5. In the **Map From** box, enter the regex of the URL's path and the query you want to match. For example: `^/bank/(.*)` This example illustrates matching the `Request-Line` in the request. The `Request-Line` begins with `/bank/` (the `^` indicates "begins with") and places the rest of the URL into the first capture group (for more information on regex patterns, see the *Oracle Java Docs*).
6. In the **Map To** box, enter the URL's path and query you want to generate. For example: `/application/$1` This example defines the replacement string, which generates `/` followed by the content of the first capture group (to better understand the use of special characters such as `\` and `$` in the replacement string, see the *Oracle Java Docs*).
7. Click **Save**.

Adding a Groovy script rule
You can add a Groovy script rule, which provides advanced rule logic that extends PingAccess rule development beyond the capabilities of the packaged *Policy Manager* rules.

About this task

> **Note:** Through Groovy scripts, PingAccess administrators can perform sensitive operations that could affect system behavior and security. Please note that since the regular Groovy Rule and the OAuth Groovy Rule differ in the scope of their functionality, the relevant rules are tagged for Web App or for API, respectively, in the rules dropdown menu.

See *Advanced Fields* for information about error handling.

Steps

1. Navigate to **Rules# Rules**.
2. Click **+ Add Rule**.

3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Groovy Script (for Web App)**.
5. Enter the Groovy Script to use for Rule evaluation. For example, to create an OAuth Scope Rule that matches more than one scope, your Groovy script might contain: `hasScopes("access","portfolio")`
6. If you need to configure error handling parameters, click **Show Advanced** to provide those configuration options.
7. Click **Save** when you finish.

Adding an HTTP Request Header rule
You can add an HTTP Request Header rule, which examines a request and determines whether to grant access to a requested resource based on a match found in one of the specified headers in the HTTP request.

About this task

Note: See *Advanced Fields* for information about error handling.

If more than one **Field** and **Value** pair is listed, then all conditions must match in order for the rule to succeed.

To configure an HTTP Request Header rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **HTTP Request Header**.
5. In the **Field** column, enter a **Header** name you want to match in order to grant or not grant the client access.
6. Enter the **Value** for the Header you want to match in order to grant or not grant the client access. The wildcard (*) character is supported.

   Tip: If you want to match on the `Host` header, include both the host and port as the **Value**, or add a wildcard after the hostname (`host*` or `host:*`) to match what is in the HTTP request.
7. Select **Case Sensitive** if the values should be matched only if the value case is an exact match.
8. Select **Negate** if access should be denied when a match is found.

   Info: Ensure that the attribute name entered in the **Field** field is spelled correctly and exists. If you enter an attribute that does not exist and you select **Negate**, the rule will always succeed. The **Negate** control applies to the entire set of conditions specified, and passes the rule if any condition is not met.
9. If additional **Header** pairs are needed, click **Add Row** to add an additional row, then repeat steps 1-4.
10. If you need to configure error handling parameters, click **Show Advanced** to provide those configuration options.
11. Click **Save**.

Adding an HTTP Request Parameter rule
You can add an HTTP Request Parameter rule, which examines a request and determines whether to grant access to a requested resource based on a match found in specified form parameters of the HTTP request.

About this task

This rule determines if the parameters are passed as part of the URL query string parameters or as part of a request body submitted using an HTTP PUT or POST method. If the request is a POST request, the `content-type` must be set to `application/x-www-form-urlencoded` to process the field names in the request.

If this rule is applied to an Agent configuration, only URL query string parameters are compared, because the Agent does not receive the request body for processing.

If more than one **Field** and **Value** pair is listed, then all conditions must match in order for the rule to succeed.

Note: See *Advanced Fields* for information about error handling.

To configure an HTTP Request Parameter rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **HTTP Request Parameter**.
5. In the **Field** column, enter a **Parameter** name you want to match in order to grant or not grant the client access.
6. Enter the **Value** for the field you want to match in order to grant or deny the client access. The wildcard (*) character is supported.

   Note: Values entered here will be URL-encoded prior to the comparison. For example, if the value specified in the **Value** field is `v1 v2`, when the engine performs the comparison, this value will be converted to `v1%20v2` before the search is performed.
7. Select **Case Sensitive** if the values should be matched if the value case is an exact match.
8. Select **Negate** if when a match is found, access is not allowed.

   Info: Ensure that the field name you enter is spelled correctly and exists. If you enter a field name that does not exist and you select **Negate**, the rule will always succeed. The **Negate** control applies to the entire set of conditions specified, and passes the rule if any condition is not met.
9. If additional **Parameters** pairs are needed, click **Add Row** to add an additional row, then repeat steps 1-4.
10. If you need to configure error handling parameters, click **Show Advanced** to provide those configuration options.
11. Click **Save**.

Adding a Network Range rule
You can add a Network Range rule, which examines a request and determines whether to grant access to a target Site based on whether the IP address falls within a specified range (using Classless Inter-Domain Routing notation).

Steps

1. Navigate to **Rules# Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Network Range**.
5. Enter a **Network Range** in the field. For example, `127.0.0.1/8`. PingAccess supports both IPv4 and IPv6 addresses.
6. Select **Negate** if when a match is found, access is not allowed.
7. If you wish to override source address handling defined in the **HTTP Requests** configuration, click **Show Advanced** and perform the following steps:
   a. Select the **Override Request IP Source Configuration** option.
   b. Enter the **Headers** used to define the Source IP address to use.
   c. Select the **Header Value Location** to use when multiple addresses are present in the specified header. Valid values are `Last` (the default) and `First`.
   d. Select the **Fall Back to Last Hop IP** option to determine if, when the specified **Headers** are not present, PingAccess should return a `Forbidden` result or if it should use the address of the previous hop as the source to make policy decisions.
8. Additional advanced fields for handling error responses may also be defined here. See *Advanced Fields* for more information about these fields.
9. Click **Save**.

Adding an OAuth Attribute Value rule
You can add an OAuth Attribute Value rule, which examines a request and determines whether to grant access to a target Service based on a match found between the attributes associated with an OAuth access token and attribute values specified in the OAuth Attribute Rule.

Steps

1. Navigate to **Rules# Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **OAuth Attribute**.
5. Select an **Attribute Name** you want to match to an attribute associated with an OAuth access token.
6. In the **Attribute Value** box, enter the value to match.

   ⓘ **Note:** The attribute values come from the contract in your OAuth access token manager in PingFederate. See *Defining the Access Token Attribute Contract* for more information.

7. Add additional rows of attribute name/value pairs as needed.

   ⓘ **Note:** If multiple rows are included here, all conditions must match in order for the rule to match.

8. Select **Negate** if when a match is found, access is not allowed.

   ⓘ **Info:** Verify what you enter for the attribute. If you enter an attribute that does not exist (for example, misspell it) and you select **Negate**, the rule will always succeed.

9. Click **Save**.

Adding an OAuth Groovy Script rule

You can add an OAuth Groovy Script rule, which determines whether to grant access to a target Site based on the results returned from a Groovy script that evaluates request details and OAuth details.

About this task

This rule allows you to create more sophisticated OAuth Scope and OAuth Attribute Value Rules for API applications.

Info: Please note that since the regular Groovy Rule and the OAuth Groovy Rule differ in the scope of their functionality, the relevant rules are tagged for Web App or for API, respectively, in the UI's rules dropdown menu.

To configure an OAuth Groovy Script rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **OAuth Groovy Script (for API)**.
5. Enter the **Groovy Script** to use for Rule evaluation.
6. If you need to specify a custom Error Response Template File, click **Show Advanced** and fill in the **Error Response Template File** and **Error Response Content Type** fields.
7. Click **Save**.

Results

See *Advanced Fields* for information about error handling.

Adding an OAuth Scope rule

You can add an OAuth Scope rule, which examines the contents of the PingFederate validation response and determines whether to grant access to a back-end target Site based on a match found between the scopes of the validation response and scope specified in the OAuth Scope Rule.

About this task

For example, a Resource may require that the OAuth Access Token contain the scope `superuser`.

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **OAuth Scope**.
5. Select the **Scope** you want to match to values returned from the access token.

   Info: This is one scope requirement in the set of scopes associated with the access token.
6. Select **Negate** if when a match is found, access is not allowed.
7. Optional: If you want to use a customized error response template, click **Show Advanced** and provide the **Error Response Template File** and **Error Response Content Type** values.
8. Click **Save**.

Adding an OAuth Token Cache Time To Live rule
You can add an OAuth Token Cache Time to Live rule, which cofigures the caching behavior for access tokens.

About this task

This rule allows the global OAuth token cache configuration to be selectively overridden for specific applications or resources.

To configure an OAuth Token Cache Time To Live rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **OAuth Token Cache Time to Live**.
5. Specify whether you want to cache the introspection of access tokens.
6. Specify the number of seconds to cache the introspection of the access token. This value should be less than the token provider's token lifetime. A value of -1 means no limit.
7. Click **Save**.

Adding a Rate Limiting rule
You can add a Rate Limiting rule, which lets you limit a client from overloading the server with too many requests in a specified period of time.

About this task

The implementation of this rule uses a *Token Bucket* in order to control the number of incoming requests.

Note: The rate limiting rule may benefit from the configuration of a PingAccess *Runtime State Cluster* to ensure rate limits are enforced properly if the front-end load balancer does not provide a sticky session.

The way this works is that the configuration defines a number of requests and an interval that must elapse between requests. The allowed number of requests within the time window is controlled by the **Max Burst Requests** setting visible when you click **Show Advanced**. For example, if the **Max Burst Requests** value is `1`, two requests are allowed in the request interval — one normal request, and one burst request.

The number of allowed requests is incremented by one at the end of each **Request Interval** if a request was not received. This continues until the number of allowed requests equals the value defined by the **Max Burst Requests** setting.

Note: Using the Rate Limiting Rule in a clustered PingAccess environment may impose stricter clock synchronization requirements for requests processed by multiple engine nodes. Alternatively, a load balancer sitting in front of a PingAccess cluster can be configured to stick the session to a specific engine, thus ensuring that the rate limiting rule is applied by a single PingAccess engine node.

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rate Limiting**.

5. Select a **Policy Granularity**, as defined in the following table:

Policy Granularity	**Definition**
Resource	Restricts the rate of requests based on the resource requested.
Identity	Restricts the rate of requests to the identity associated with the current authentication token (a PA Cookie or an OAuth token). This is the default value.
IP	Restricts the number of requests based on the source IP address. The IP address used to apply this policy comes from the HTTP Requests IP Source configuration options or options that override that configuration, if those options are configured.
OAuth Client	Restricts the number of requests to all OAuth tokens obtained by a specific **Client ID**.

6. Enter, in milliseconds, a **Request Interval**.
7. If more than `1` request should be allowed a request interval, expand the **Advanced** section of the page, and enter the number of requests to allow in the **Max Burst Requests** field.

> **Note:** PingAccess increases the number of available requests only after a request interval that serves no requests to the client. As a result, in the period following a cycle where the remaining allowed burst requests is reduced to `0`, no burst requests would be allowed, regardless of this setting.

8. If PingAccess should reply to the client with a `Retry-After` header instructing the client to wait for a period of time, select the **Set Retry-After Header** option.
9. To customize the error response sent to the client, click **Show Advanced** and modify the **Error Response Code**, **Error Response Status Message**, **Error Response Template File**, and **Error Response Content Type** fields. See *Advanced Fields* for more information about these fields.
10. Click **Save**.

Adding a Redirect rule

You can add a Redirect rule, which specifies a fixed URL that a user agent should request in response to being denied access to the requested resource.

About this task

Redirect rules can be applied to one or more applications.

To configure a Redirect rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Redirect**.
5. Specify the **Response Status Code** you want to associate with the redirect. The default is `302`.
6. Specify the **URL** to which you want to redirect requests. URLs must include the `http/https` prefix. The URL can be specified with or without defining the port.
7. Click **Save**.

Adding a Rejection rule
You can add a rejection rule, which specifies an action to take when a request to an application or resource is rejected by policy evaluation.

About this task

A rejection rule uses Rejection Handlers to define which action you want to take, either rejecting the request and displaying an error template, or redirecting the user to another URL for error details, instructions, or additional actions.

To configure a **Rejection** rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Rejection**.
5. Specify the **Rejection Handler** you want to use for the rule.
6. Click **Save**.

Adding a Time Range rule
You can add a Time Range rule, which examines a request and determines whether to grant access to a back-end target site based on the request falling within a defined time frame.

About this task

For example, use this Rule when you want to restrict access to specific endpoints for certain time periods, such as during the work day from 8 am to 5 pm.

To configure a Time Range rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Time Range**.
5. Enter the beginning time for the time frame in the **Start Time** field. For example: 8:00 AM.
6. Enter the ending time for the time frame in the **End Time** field. For example: 5:00 PM.

> **Info:** If you are using Internet Explorer or Firefox, you must enter the time in 24 hour format. For example, 5:00 PM is 17:00.

7. Select **Negate** if when a match is found, access is not allowed.
8. Click **Save** when you finish.

Results

See *Advanced Fields* for information about error handling.

Adding a Web Session Attribute rule
You can add a Web Session Attribute rule, which examines a request and determines whether to grant access to a target Site based on an attribute value match found within the PA Token.

Steps

1. Navigate to **Rules# Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Web Session Attribute**.
5. Select the **Attribute Name** that you want to match in order to grant the client access. For example, `Group`.
6. Enter the **Attribute Value** for the Attribute Name. For example, `Sales`. If the attribute has multiple values at runtime, the attribute value you specify here must match one of those values.

   Info: PA Token attributes are obtained from the PingFederate OpenID Connect Policy attribute contract (see *Configuring OpenID Connect Policies*).

7. Click **Add Row** to add more attributes, or click to remove a row.
8. Select **Negate** to disallow access when a match is found.

   Info: Ensure the attribute name is spelled correctly and exists. If you enter an attribute that does not exist and you select **Negate**, the rule will always succeed.

9. Click **Save**.

Results

Info: To use this rule, we recommend that you leave the **Request Profile** checkbox selected, indicating that you want PingAccess to request additional profile attributes from PingFederate when requesting the ID Token.

See *Advanced Fields* for information about error handling.

Adding a Web Session Scope rule
You can add a Web Session Scope rule, which examines the contents of the PingFederate validation response and determines whether to grant access to a back-end target site based on a match found between the scopes of the validation response and scope specified in the Web Session Scope Rule.

About this task

For example, a Resource may require that the Access Token contain the scope `superuser`.

To configure PingFederate to support this rule:

1. *Enable Expressions*.
2. *Extend the Access Token Attribute Contract* to include the value `scope`.
3. Map the following value into the *access token attribute contract*:

Contract	Source	Value
scope	Expression	@com.pingidentity.sdk.oauth20.Scope@encod

4. Manage the *OpenID Connect policy* to add the following information:
   - *Attribute Contract* - Extend the contract to include the `scope` attribute. Choose to **Override Default Delivery** using the **ID Token**.

     ⓘ **Note:** This step is not applicable to PingFederate 9.0 and earlier. Instead, on the **Manage Policy** screen, select the check box to **Include User Info in ID Token**.
   - *Attribute Scopes* - Associate the `openid` **Scope** with the `scope` **Attribute**.

     ⓘ **Note:** This feature does not exist in PingFederate versions prior to 9.0. To work around this issue:

     a. Ensure PingAccess is configured to include `profile` in the list of Web Session scopes.
     b. In PingFederate, ensure the `profile` scope is defined in *Scope Management*.
     c. During authentication, the user must accept usage of the `profile` scope. If the user does not accept usage of profile scope then the Web Session Scope rule will always fail for that user.
   - *Contract Fulfillment* - Modify the `scope` **Attribute Contract** to use `Access Token` as the **Source** with a **Value** of `scope`.

To configure a Web Session Scope rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **Web Session Scope**.
5. Select the **Scope** you want to match to values returned from the access token.

   ⓘ **Info:** This is one scope requirement in the set of scopes associated with the access token.
6. Specify the **Rejection Handler** you want to associate with this rule.
7. Click **Save**.

Adding a WebSocket Handshake rule
You can add a WebSocket handshake rule, which lets you define the domains that can open a cross-origin WebSocket to the application or resource.

About this task

You can also define allowed WebSocket subprotocols and extensions, providing more fine-grained control over how the application behaves.

To configure a WebSocket Handshake rule:

Steps

1. Navigate to **Rules**# **Rules**.
2. Click **+ Add Rule**.
3. Enter a unique **Name**. The name can be up to 64 characters long. Special characters and spaces are allowed.
4. In the **Type** dropdown, select **WebSocket Handshake**.

5. Enter one or more **Allowed Origins**. If no origins are defined, all cross-origin WebSocket requests are denied.

> **Important:** While it is allowed, we recommend against using a value of `*` in this field. While this is a valid configuration, it is considered to be an insecure practice.

6. Modify the list of **Allowed Subprotocols**. Subprotocols are defined in the `Sec-WebSocket-Protocol` handshake header. The default value of `*` allows all subprotocols.
7. Modify the list of **Allowed Extensions**. WebSocket extensions are defined in the `Sec-WebSocket-Extensions` handshake header. The default value of `*` allows all extensions.
8. Click **Save**.

Results

See *Advanced Fields* for information about error handling.

Editing a rule
You can edit the properties of an existing rule.

Steps

1. Navigate to **Rules**# **Rules**.
2. Expand the rule you want to edit and click .
3. Make the required changes.
4. Click **Save**.

Deleting a rule
You can delete an existing rule.

Steps

1. Go to **Rules**# **Rules**.
2. Expand the rule you want to delete and click .
3. When prompted, click **Delete** to delete the rule.

Manage Rule Sets
Rule sets let you combine rules into reusable groupings.
Adding a rule set
You can add a new rule set.

Steps

1. Go to **Rules**# **Rule Sets**.
2. Click **+ Add Rule Set**.
3. Drag a Rule from the **Rules** column onto the box that appears.
4. Enter a name for the Rule Set in the box that appears. Special characters and spaces are allowed.

5. Select **All** as the **Success Criteria** to require all Rules in the set to succeed. Select **Any** to require just one of the Rules in the set to succeed.

> ⓘ **Info:** When **Success Criteria** is set to **Any**, the first rule establishes the error handling and is flagged with a tooltip that displays a message indicating that. When **Success Criteria** is set to **All**, the first rule in the set that fails establishes the error handling.

> ⓘ **Note:** When **Success Criteria** is set to **Any**, PingAccess flags Processing Rules in a Rule Set with tooltip that warns that if the first rule in the list succeeds, additional rules will not be processed. This is considered a misconfiguration because in an **Any** Rule Set, the first Processing Rule should succeed, causing all other rules in the set to not be evaluated. If you want to use Processing Rules on protected applications as well as handle access control decisions using **Any** criteria, assign Processing Rules directly to the application or create a separate Rule Set for the Processing Rules using the **All** criteria."

6. Click **Save** to save the Rule Set.
7. Add more Rules.

Editing a rule set
You can edit an existing rule set.

Steps

1. Navigate to **Rules**# **Rule Sets**.
2. Expand the rule set you want to edit. Choose from:
   - Drag a rule within a rule set up or down to re-order the rules.
   - Click **-** to the right of any rule you want to remove from a rule set.
   - Click 🗑 to delete the rule set.
   - Click ✎ to edit the rule set **Name** or **Success Criteria**.
3. Click **Save** after changing these values.

Deleting a rule set
You can delete an existing rule set. You must remove any associations between the rule set and any applications or resources before you can delete it.

Steps

1. Navigate to **Rules**# **Rule Sets**.
2. Expand the rule set you want to delete.
3. Click 🗑.
4. When prompted, click **Remove** to confirm the delete request.

Manage Rule Set Groups
Rule set groups let you combine one or more rule sets into reusable groups.
Adding a rule set group
You can add a new rule set group.

Steps

1. Navigate to **Rules**# **Rule Set Groups**.
2. Click **+ Add Rule Set Group**.
3. Drag a Rule Set from the **Rule Sets and Groups** column onto the box that appears.
4. Enter a name for the Rule Set Group in the box that appears. Special characters and spaces are allowed.

5. Select **All** as the **Success Criteria** to require all Rules in the set to succeed. Select **Any** to require just one of the Rules in the set to succeed.

> **Info:** When **Success Criteria** is set to **Any**, the first rule set establishes the error handling and is flagged with a tooltip that displays a message indicating that. When **Success Criteria** is set to **All**, the first rule in the set that fails establishes the error handling.

> **Note:** When **Success Criteria** is set to **Any**, PingAccess flags Processing Rules in a Rule Set Group with tooltip that warns that if the first rule in the list succeeds, additional rules will not be processed. This is considered a misconfiguration because in an **Any** Rule Set Group, the first Processing Rule should succeed, causing all other rules in the set to not be evaluated. If you want to use Processing Rules on protected applications as well as handle access control decisions using **Any** criteria, assign Processing Rules directly to the application or create a separate Rule Set Group for the Processing Rules using the **All** criteria."

6. Click **Save** to save the Rule Set Group.
7. Add more rule sets or rule set groups.
8. Configure rule set hierarchy by dragging rule sets in the rule set group. Processing occurs from top to bottom. The configuration is automatically saved.

Editing a rule set group
You can edit the properties of an existing rule set group.

Steps

1. Navigate to **Rules**# **Rule Sets and Groups**.
2. Expand the rule set group you want to edit. Chose from:
   - Drag a rule set within a rule set group up or down to re-order the rules.
   - Click **-** to the right of any rule set you want to remove from a rule set group.
   - Click to delete the rule set group.
   - Click to edit the rule set group**Name** or **Success Criteria**.
3. Click **Save** after changing these values.

Deleting a rule set group
You can delete an existing rule set group. You must remove any associations between the rule set group and any applications or resources before you can delete it.

Steps

1. Go to **Rules**# **Rule Set Groups**.
2. Expand the rule set group you want to delete.
3. Click .
4. When prompted, click **Remove** to confirm.

Manage Rejection Handlers
A rejection handler defines the action to take when a request to an application or resource is rejected by policy evaluation. This lets you decide if you want to display an **error template** or **redirect** the user to another URL for error details, instructions, or additional actions.

You can specify the response status code that you want to send if policy evaluation fails.

You include a rejection handler in a rule by expanding the **Advanced** section of the **Create Rule** screen.

PingAccess contains 3 predefined rejection handlers:

- Default API Rejection Handler – Returns a 403 status code in a JSON template

- Default Rate Limiting Rejection Handler – Returns a 429 status code in a JSON template
- Default Web Rejection Handler – Returns a 403 status code in HTML template

Creating a rejection handler
You can create a new rejection handler.

Steps

1. Navigate to **Rules**# **Rejection Handlers**.
2. Click **+ Add Rejection Handler**.
3. Specify a **Name** for the object.
4. Choose the **Type**, **Error Template** or **Redirect**.
   - If you selected Error Template, specify the **Response Status Code**, the **Template File**, and the **Content Type**.
   - If you selected Redirect, specify the **Response Status Code**, and **URL** to which you want to redirect if policy evaluation fails.

Editing a rejection handler
You can edit the properties of an existing rejection handler.

Steps

1. Go to **Rules**# **Rejection Handlers**.
2. Expand the rule you want to edit and click .
3. Make the required changes.
4. Click **Save**.

Deleting a rejection handler
You can delete an existing rejection handler.

Steps

1. Go to **Rules**# **Rejection Handlers**.
2. Expand the rule you want to delete and click .
3. When prompted, click **Remove** to delete the rule.

Access

The **Access** tab contains controls for authentication requirements, identity mappings, virtual hosts, web sessions, and unknown resources.

Choose from one of the following sections:

- *Authentication requirements* on page 182
- *Identity Mappings* on page 183
- *Virtual Hosts* on page 186
- *Web Sessions* on page 188
- *Unknown Resources* on page 193

Authentication requirements
Authentication requirements are policies that dictate how a user must authenticate before access is granted to a protected Web Application.

Authentication methods are string values and ordered in a list by preference. At runtime, the type of authentication attempted is determined by the order of the authentication methods.

For example, a user attempts to access a PingAccess Web Application configured with an authentication requirement list containing the values (password, cert). PingAccess redirects the user to PingFederate requesting either password or certificate user authentication. PingFederate authenticates the user based on the password and issues an OIDC ID Token to PingAccess (containing the authentication method that was used). PingAccess ensures that the authentication method matched the requirements and redirects the user to the originally requested Application with the PA cookie set. The user navigates to the Application and access is granted. When the user attempts to access a more sensitive Application, configured with an authentication requirement list containing the value (cert), they are redirected to PingFederate to authenticate with a certificate.

If you configure Applications with authentication requirement lists that have no overlap. For example, one list has (password), another list (cert), a user navigating between Applications may be required to authenticate each time they visit an Application. When configuring authentication requirement lists to protect higher value Applications with step-up authentication, consider including stronger forms of authentication when configuring lower value Applications.

Configuring an authentication requirements list
You can configure a list of authentication requirements.

Steps

1. Navigate to **Access# Authentication Requirements**, and click **Add Authentication Requirement**.
2. Enter a unique name for the Authentication Requirements list.
3. Enter an authentication method. For example, `cert` or `password`.

> **Info:** The values you enter here must match the result values defined for the Requested AuthN Context Selector configured within PingFederate (see *Configuring the Requested AuthN Context Selector*).

4. Click **Add Authentication Requirement** to add additional authentication requirements.
5. Click **Save**.

Editing an authentication requirements list
You can edit the properties of an existing authentication requirements list.

Steps

1. Navigate to **Access# Authentication Requirements**.
2. Expand list you want to edit, then click .
3. Make your changes.
4. Click **Save**.

Deleting an authentication requirements list
You can delete an existing authentication requirements list.

Steps

1. Navigate to **Access# Authentication Requirements**.
2. Expand the authentication requirement you want to delete.
3. Click .
4. When prompted, click **Delete** to confirm.

Identity Mappings
Identity mappings make user attributes available to back-end sites that use them for authentication. There are multiple types of identity mappings, each with different behavior and a distinct set of fields to specify the identity mapping behavior.

Configuring Auth Token Management

You can configure auth token management to define the issuer and signing configuration used by JWT Identity Mappings.

Steps

1. Navigate to **Access# Identity Mappings**.
2. Specify the **Key Roll Interval (h)** to indicate how often (in hours) you want to roll the keys. Key rollover updates keys at regular intervals to ensure the security of the signed auth tokens.
3. Specify a published, unique **Issuer** identifier to be used with auth tokens. For example, set the issuer to a value that more closely represents your company. PingAccess inserts this value as the iss claim within the auth tokens.
4. Specify the **Signing Algorithm** used to protect the integrity of the auth tokens (the default is ECDSA using P-256 Curve).
5. Click **Save**.

Creating a header identity mapping

You can create a header identity mapping to make user attributes or client certificates available as HTTP request headers to applications, both site and agent based, that use them for authentication.

About this task

A single Header identity mapping can expose a number of attribute values or a certificate chain up to 3 levels deep. Header identity mappings are assigned to applications.

Steps

1. Navigate to **Access# Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. Enter a **Name** for the mapping.
4. In the **Type** dropdown, select **Header Identity Mapping**.
5. Optional: In the `Attribute to Header Mapping` section, enter the name of the attribute to retrieve from the user web session in the **Attribute Name** field. For example, `sub`.
6. Enter the name of the HTTP requests header to contain the attribute value in the **Header Name** field. The HTTP header you specify here is the actual header name over the HTTP protocol, not an environment variable interpreted format. For example, enter the `User-Agent` browser type identifying header as `User-Agent`, not `HTTP_USER_AGENT`.
7. Select which attribute is used as the **Subject**.
8. Optional: In the `Certificate to Header Mapping` section, enter the header name to contain a PEM-encoded client certificate. The row position correlates to the index in the client certificate chain. For example, the first row always maps to the leaf certificate. If you are using a certificate chain, click **Add Row** to add another row.
9. Click **Save**.

Creating a JWT identity mapping

You can create a JWT Identity Mapping to make user attributes available in a signed JWT (JSON Web Token) sent to the application in a header.

About this task

The JWT issuer and signing configuration is defined in *Configuring Auth Token Management* on page 184.

When configuring identity mappings, the dot notation is supported so that session token structure can be maintained. For example, if the session token contains the following entry:

```
{
  "address": {
     "line1": "123 Any St",
     "line2": "Apt 123",
     "city": "Anytown",
     "state": "CO",
     "zip": "12345"
   }
}
```

You can define an identity mapping using the following entries to maintain the structure of the target JWT:

User Attribute Name	JWT Claim Name
address.line1	address.line1
address.line2	address.line2
address.city	address.city
address.state	address.state
address.zip	address.zip

Tip: PingAccess engines provide a JWKS (JSON Web Key Set) endpoint at /pa/authtoken/JWKS that can be used by backend sites to validate the signature of the JWT.

Steps

1. Navigate to **Access**# **Identity Mappings**.
2. Click **+ Add Identity Mapping**.
3. Enter a **Name** for the mapping.
4. In the **Type** dropdown, select **JWT Identity Mapping**.
5. Enter the name of the header to use when sending the signed JWT to the target application in the **Header Name** field. The HTTP header you specify here is the actual header name over the HTTP protocol, not an environment variable interpreted format. For example, enter the `User-Agent` browser type identifying header as `User-Agent`, not `HTTP_USER_AGENT`.
6. Enter the audience to be set as the `aud` claim in the signed JWT in the **Audience** field.
7. In the Attributes section, select a list type. An inclusion list includes only the specified attributes, and an exclusion list includes all attributes not specified.
8. If you selected an inclusion list, configure the inclusion list:
    a. Enter the name of the attribute to retrieve from the user web session in the **User Attribute Name** field. For example, `sub`.
    b. Enter the name of the JWT claim to contain the attribute value in the **JWT Claim Name** field.
    c. Select which included attribute is used as the **Subject**.
9. If you selected an exclusion list, configure the exclusion list:
    a. Enter the names of the attributes to exclude.
    b. Select which included attribute is used as the **Subject**.
10. Optional: In the **Certificate to JWT Claim Mapping** field, enter the name of the JWT claim to contain the client certificate chain array.

11. **Client Certificate Chain Max Depth**If you are performing Certificate to JWT Claim Mapping, in the field, specify the maximum number of certificates from the client certificate chain included in the JWT claim array.
12. Optional: Click **Show Advanced** and select **Cache JWTIf you are performing Certificate to JWT Claim Mapping, in the** . When enabled, a cached signed JWT will be used for repeated requests for a given user. If user attributes change or the key used to sign the JWT changes, a new JWT will be created even if JWT caching is enabled.
13. Click **Save**.

Editing an identity mapping
You can edit the properties of an existing identity mapping.

Steps

1. Navigate to **Access**# **Identity Mappings**.
2. Expand the identity mapping you want to edit.
3. Click ✎.
4. Make the required changes.
5. Click **Save**.

Deleting an identity mapping
You can delete an existing identity mapping.

Steps

1. Navigate to **Access**# **Identity Mappings**.
2. Expand the identity mapping you want to edit.
3. Click the Delete icon.
4. In the confirmation window that appears, click **Delete**.

Virtual Hosts
Virtual Hosts enable PingAccess to protect multiple application domains and hosts.

A Virtual Host is defined by the host name and host port.

A wildcard (`*`) can be used either to define either any host (`*:443`, for example) or any host within a domain (`*.example.com`, for example). If a request matches more than one virtual host, the most specific match is used.

ⓘ **Note:** Prior to availability of SNI in Java 8, an HTTPS port could only present a single certificate. In order to handle multiple Virtual Hosts you have to use a wildcard name certificate or the *Subject Alternative Name* (SAN) extension. With SNI available, Virtual Hosts can present different certificates on a single HTTPS port. You can assign which certificates (Key Pairs) are used by which Virtual Host on the *HTTPS Listeners* page.

The Agent Resource Cache TTL advanced field is used to control PingAccess Agent resources for each virtual host.

If you configure a trusted certificate group for a virtual host, or configure an engine key pair to associate it with a virtual host, those settings are used instead of any applicable HTTPS listeners or engine listeners for the virtual host.

Creating a new virtual host
You can create a new virtual host.

Steps

1. Navigate to **Access# Virtual Hosts**.
2. Click **Add Virtual Host**.
3. Enter the **Host** name for the Virtual Host. For example: myHost.com. You can use a wildcard (*) to indicate that any host name is acceptable. A wildcard host may also be specified (e.g. `*.example.com`).
4. Enter the **Port** number for the Virtual Host. For example: `1234`.
5. Enter the **Agent Resource Cache TTL** indicating the number of seconds the Agent can cache resources for this application. Only applies to destination of type `Agent`.
6. Click **Save**.

Configuring virtual host trusted certificate groups
You can configure a virtual host trusted certificate group that can then be used to implement client certificate authentication.

About this task

Assigning a trusted certificate group to a virtual host provides a mechanism to authenticate using client certificates during any request to sites using the specified virtual host.

Note: Trusted certificate groups are applied at the host name level and are independent of the configured port. This means that a mapping to a virtual host of `*.example.com` will apply to requests received on virtual hosts `*.example.com:3000` and `*.example.com:443`.

Steps

1. Navigate to **Access# Virtual Hosts**. Virtual Hosts that have certificate authentication configured will display the message *Client Certificate Authentication* in the associated bar.
2. Select and expand the Virtual Host you want to modify.
3. In the **Properties** list, click the ✎ to select a **Trusted Certificate Group**.
4. Select the appropriate **Trusted Certificate Group**.
5. Click **Save**.
6. Add the following two *Groovy script rules* to force validation of the SNI and client certificate chain. *Apply these rules* to applications using this virtual host.

   `Validate SNI`

   ```
   if(exc?.getSslData()?.getSniServerNames()?.isEmpty())
   {
     fail();
   }
   else
   {
     pass();
   }
   ```

   `Validate client certificate chain`

   ```
   if(exc?.getSslData()?.getClientCertificateChain()?.isEmpty())
   {
     fail();
   }
   else
   {
     pass();
   ```

```
}
```

Editing a virtual host
You can edit the properties of an existing virtual host.

Steps

1. Navigate to **Access**# **Virtual Hosts**.
2. Expand the virtual host you want to edit and click ✎.
3. Make the required changes.
4. Click **Save**.

Deleting a virtual host
You can delete an existing virtual host.

Steps

1. Navigate to **Access**# **Virtual Hosts**.
2. Expand the virtual host you want to edit and click 🗑.
3. Click **Delete** to confirm.

Web Sessions
Web sessions define the policy for web application session creation, lifetime, timeouts, and their scope.

You can configure any number of web sessions to scope the session to meet the needs of a target set of applications. This improves the security model of the session by preventing unrelated applications from impersonating the end user. Use the following tasks to configure secure Web Sessions for use with specific applications and to configure global Web Session settings.

Application scoped Web Sessions

PingAccess Tokens can be configured to have their Web Sessions scoped to a specific application. This improves the security model of the session by preventing unrelated applications from impersonating the end user.

Several controls exist to scope the PA Token to an application:

Audience Attribute

The audience attribute defines who the token is applicable to and is represented as a short, unique identifier. Requests are rejected that contain a PA Token with an audience that differs from what is configured in the Web Session associated with the target Resource.

Audience Suffix

The audience attribute is also used as a suffix of the cookie name to ensure uniqueness. For example, PA.businessAppAudience.

Cookie Domain

The cookie domain can also optionally be set to limit where the PA Token is sent.

ⓘ **Info:** In addition to these controls, parameters such as session timeout can be adjusted to match the policy requirements of each application.

Corresponding OAuth clients must be defined in PingFederate for each Web Session. Redirect URL whitelists defined in PingFederate dictate from which servers and domains the session can originate. Controlling this within PingFederate enables flexibility of the attribute contract (and its fulfillment) for that particular application. This ensures that each application and its associated policies only deal with attributes related to it.

Configure Web Session Management settings
You can configure web session management sessions.

Steps

1. Navigate to **Access**# **Web Sessions**.
2. In the **Web Session Management** section, enter the **Key Roll Interval**, in hours, to specify how often you want to roll the keys (the default is `24` hours). Key rollover updates keys at regular intervals to ensure the security of signed and encrypted PA Tokens.
3. In the **Issuer** field, enter the published, unique identifier to be used with the Web session (the default is PingAccess). For example, set the issuer to a value that more closely represents your company. PingAccess inserts this value as the `iss` claim within the PA Token.
4. Select the **Signing Algorithm** used to protect the integrity of the PA Token (the default is `ECDSA using P-256 Curve`). PingAccess uses the algorithm when creating signed PA Tokens and when verifying signed tokens in a request from a user's browser. The algorithm is also used for signing tokens in Token Mediation use cases when PA Tokens are encrypted.
5. Select the **Encryption Algorithm** used to encrypt and protect the integrity of the PA Token (the default is `AES 128 with CBC and HMAC SHA 256`). PingAccess uses the algorithm when creating encrypted PA Tokens and when verifying them from a user's browser.

   Info: Higher encryption levels are available if the administrative console supports it. To enable higher encryption levels, update the administrative console JRE to support unlimited strength security policy.

   Info: In a clustered environment, be sure to add the security policy changes to the engines as well as the administrative console for the cluster.
6. Enter the browser **Cookie Name** that contains the PA Token (the default is `PA`).
7. In the **Session State Cookie Name** field, enter a name for the browser cookie to contain session state attributes.
8. In the **Update Token Window(s)** field, enter the number of seconds before the idle timeout is updated in the PA token. When this time window expires, PingAccess will reissue a new PA cookie.
9. In the **Nonce Cookie Time to Live (M)** field, enter the number of minutes for which the nonce cookie is valid. The default value is `5`. PingAccess deletes cookies that are older than this threshold.
10. Click **Save**.

Creating a Web Session
You can create a new web session.

Steps

1. Go to **Access**# **Web Sessions**
2. On the Web Session page, click **+ Add Web Session**.
3. Enter a unique **Name** for the web session, up to 64 characters, including special characters and spaces.
4. Select a **Cookie Type**.

   An **Encrypted JWT** token uses authenticated encryption to simultaneously provide confidentiality, integrity, and authenticity of the PA Token.

   A **Signed JWT** token uses asymmetric cryptography with a private/public key pairing to verify the signed message and to confirm that the message was not modified during transit.

   Encrypted JWT is the default setting. Changing this setting may affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.

5. Specify the **Audience** that the PA Token is applicable to, represented as a short, unique identifier between 1 and 32 characters.

   Requests are rejected that contain a PA Token with an audience that differs from what is configured in the Web Session associated with the target application. Changing this setting may affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.
6. Specify the **OpenID Connect Login Type** that defines how the user's identity is verified based on authentication performed by an OpenID Provider and how additional profile claims are obtained. Three login profiles are supported: **Code** and **POST**, and **x_post**. Select a login profile.

   Code

   A standard OpenID Connect login flow that provides confidentiality for sensitive user claims. In this profile the relying party (PingAccess) makes multiple back-channel requests in order to exchange an authorization code for an ID Token and then exchange an access token for additional profile claims from the UserInfo endpoint at the provider (PingFederate). This login type is recommended for maximum security and standards interoperability.

   POST

   A login flow that uses the `form_post` response mode. This flow follows the *OAuth 2.0 Form Post Response Mode* draft specification. This option requires PingFederate 7.3 or later.

   A form auto-POST response containing the ID Token (including profile claims) is sent to PingAccess from PingFederate via the browser after authentication. Back-channel communication between PingAccess and PingFederate is required for key management in order to validate ID Tokens. This login type is recommended for maximum performance in cases where the exchanged claims do not contain information that should be hidden from the end user.

   Be sure to select the **Implicit** grant type when configuring the OAuth Client within PingFederate (see *Configuring a Client*). The ID Token Signing Algorithm in PingFederate must be set to either one of the ECDSA algorithms or one of the RSA algorithms.

   x_post

   A login flow based on OpenID Connect that passes claims from the provider via the browser. As with the **POST** login type, select the **Implicit** grant type and use either one of the ECDSA algorithms or one of the RSA algorithms as the ID Token Signing Algorithm.

   Info: If PingFederate 7.3 or later is used in the environment, we recommend using **POST** rather than **x_post**, as **x_post** was defined by Ping Identity prior to the development of the OAuth 2.0 Form Post Response Mode draft specification.
7. Specify the **Client ID** that was assigned when you created the OAuth Relying Party client within PingFederate (for more information, see *Configuring a Client* in the PingFederate documentation). Enter the unique identifier (Client ID).
8. Specify the **Client Secret** that was assigned when you created the OAuth Relying Party Client within PingFederate. Required when configuring the **Code** Login Type. Enter the secret (Client Secret).

   Info: The OAuth Client you use with PingAccess Web sessions must have an OpenID Connect policy specified (for more information see *Configuring OpenID Connect Policies* in the PingFederate documentation).
9. Specify an **Idle Timeout** that defines the amount of time, in minutes, that the PA Token remains active, when no activity is detected by the user (the default is `60` minutes). Enter, in minutes, the length of time you want the PA Token to remain active when no activity is detected. Defining an idle expiration protects against unauthorized use of the resource by limiting the amount of time the session remains active when not being used. For example, idle expiration is useful when a user is no longer

at the computer and does not log out of the session. When the idle expiration is reached, the session automatically terminates.

> ⓘ **Info:** If there is an existing valid PingFederate session for the user, an idle time out of the PingAccess session may result in its re-establishment without forcing the user to log in again.

10. Specify a **Max Timeout** that defines the maximum amount of time, in minutes, that the PA Token remains active (the default is `240` minutes). Enter, in minutes, the length of time you want the PA Token to remain active. Once the PA Token expires, an authenticated user must re-authenticate. This protects against unauthorized use of a resource, ensuring that a session ends after the specified time and requiring the user to re-authenticate to continue.

> ⓘ **Note:** This value needs to be set to a smaller value than the PingFederate Access Token Lifetime defined in the PingFederate Access Token Management instance. See *Configuring Reference-Token Management* in the *PingFederate Administrator's Manual* for more information.

11. To configure advanced settings, click **Show Advanced**.

12. Specify the valid **Cookie Domain** where the cookie is stored. For example, `corp.yourcompany.com`.

> ⓘ **Info:** If you set the Cookie Domain, all of your web resources must reside within that domain. If you do not set the Cookie Domain, the PA Token is recreated for each host domain where you access applications.

13. Select **Secure Cookie** to indicate that the PingAccess cookie must be sent using only HTTPS connections. Selected by default.

> ⓘ **Note:** Setting an invalid Cookie Domain or selecting **Secure Cookie** in a non-HTTPS environment causes authentication to fail. This results in PingAccess re-directing the user to re-authenticate with PingFederate indefinitely.

14. Select **HTTP-Only Cookie** to enable the `HttpOnly` flag on cookies that contain the PA Token. An `HttpOnly` flagged cookie is not accessible using non-HTTP methods such as calls via JavaScript (for example, referencing `document.cookie`) and therefore cannot be easily stolen via cross-site scripting.

15. In the **SameSite Cookie** dropdown, select the level of restriction for when cookies may be sent in a cross-site request. The options are:

- Lax – The cookie should be sent on the initial navigation to a site, and is sent in same-site requests but not cross-site requests.
- None – The cookie is intended to be used across different sites without restriction.
- Disabled – The SameSite attribute is not set. This option is the default.

> ⓘ **Note:** Safari 12 will not function correctly if the SameSite attribute is set to None. Regardless of the selected setting, the SameSite attribute is disabled if Safari 12 is detected.

> ⓘ **Note:** See the *Known Issues and Limitations* for information about a browser issue that can prevent login if the SameSite Cookie attribute is set.

16. Configure the **Scopes** you want to request from the token provider when requesting the ID token. If you have a token provider configured, published scopes are available to select from the list based on the selected Client ID. You can specify unverified scopes by typing the scope and clicking **Use unverified scope "[scopename]"**.

Your token provider must be properly configured to handle all of the requested scopes you specify, including any custom scope values.

Note: The user can access all attributes by examining browser traces. While they are integrity protected to prevent changes, any sensitive or confidential attributes can be viewed should the user decode the ID Token's value.

17. Select **Validate Session** so that PingAccess will validate sessions with the configured PingFederate instance during request processing. Use of this feature requires additional configuration in PingFederate. This option is not selected by default.

    Session timeouts are synchronized between PingAccess and PingFederate when the following conditions are met:

    - A minimum release of PingFederate 8.2 is deployed with **Authentication Session Settings** configured.
    - You have selected the **Validate Session** checkbox.

    Changing this setting may affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.
18. When **Refresh User Attributes** is enabled, PingAccess will periodically contact PingFederate to update user data used in evaluating policy claims. This option works in conjunction with the PingAccess Web Session Management features to automatically require user re-authentication if user attribute data used as issuance criteria for a token in PingFederate causes the token to be revoked.

    For example, if the PingFederate OpenID Connect Policy has issuance criteria configured to only issue a token if the account is enabled, enabling this Web Session option allows PingAccess to terminate the session the next time the user accesses a protected resource if the user's account was disabled in the user datastore.

    The **Refresh User Attributes Interval** determines the length of time the user data is cached, so the effect of a change that results in a session being terminated may take up to 60 seconds (by default) to take effect. This interval can be tuned by adding `pa.websession.refreshSessionInterval` to `conf/run.properties` and assigning it a value in seconds.

    Changing this setting may affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.

    This option is selected by default.
19. When **Cache User Attributes** is enabled, PingAccess caches user attributes internally for use in policy decisions. By doing this, an attribute list that is longer than the maximum cookie size can contain information used to evaluate access requests. In practice, this is 4096 bytes, although the maximum cookie size can vary depending on the browser.

    When this option is disabled, user attribute data is encoded, signed or encrypted (depending on the web session cookie type), and stored in the browser's cookie store. The information is sent from the browser back to PingAccess with each request.

    Changing this setting may affect existing ongoing sessions, forcing the user to re-authenticate to access protected resources.

    This option is not selected by default.

20. Specify a **Consult Server Duration** to define the maximum amount of time, in seconds, that a PingAccess Agent caches policy decisions for the web session before sending a request to the Policy Server. This option only applies to agents.

> **Info:** The value used for this setting should not be larger than the **Idle Timeout** value, and ideally, should be defined to be a value less than half the timeout.

21. Select **Request Preservation** to specify the type of request data to be preserved if the user is redirected to an authentication page when submitting information to a protected resource. Available options are **None**, **POST**, or **POST and Fragment**.
22. Specify the type of **Web Storage** to be used for request preservation data.

    Session Storage is recommended. Use **Local Storage** if it is common for users to use Internet Explorer with security zones enabled and PingFederate is in a different zone than PingAccess.
23. Click **Save**.

Editing a Web Session
You can edit the properties of an existing web session.

Steps

1. Navigate to **Access**# **Web Sessions**.
2. Expand the Web Session you want to edit and click ✎.
3. Edit the Web Session.
4. Click **Save**.

Deleting a Web Session
You can delete an existing web session.

About this task

> **Info:** If the Web Session is currently associated with an application, you cannot delete it.

Steps

1. Navigate to **Access**# **Web Sessions**.
2. Expand the web session you want to delete.
3. Click 🗑.
4. When prompted, click **Delete** to confirm the request.

Unknown Resources
Unknown resources are those for which there is no associated application.

These settings define the error responses to be generated for requests that don't match the virtual host and context root of an application. Additionally, agents may be configured to allow unprotected access instead of returning an error response.

Configuring unknown resource management
You can define the action to take when an unknown resource - a resource with no matching application - is requested.

Steps

1. Navigate to **Access**# **Unknown Resources**.
2. Specify the **Error Status Code** for the HTTP response. This must be a client/server error code in the range 400 - 599.

3. Specify the name of the velocity **Error Template File** to use for generating the response body. This template file is located in the `PA_HOME/conf/template/` directory.
4. Specify the **Error Content Type** of the response; HTML, JSON, TEXT, or XML.
5. To audit unknown resource activity, select the **Audit** check box.
6. Specify the **Agent Default Mode** that determines whether an agent should **Deny** requests for unknown resources and generate an error response or allow requests to **Pass-Through** unfiltered. This default setting may be overridden by individual agents.
7. Specify the default agent resource **Cache TTL** (in seconds) to be used for unknown resources if **Pass-Through** mode is enabled.
8. Click **Save**.

Networking

The **Networking** tab controls how PingAccess manages network requests and load balancing.

Choose from one of the following sections:

- *Availability Profiles* on page 194
- *HTTP Requests* on page 195
- *Listeners* on page 196
- *Load Balancing Strategies* on page 198
- *Proxies* on page 199

Availability Profiles

Availability Profiles are used in a Site configuration to define how PingAccess classifies a backend target server as failed. Sites require the selection of an availability profile, even if only one target is provided.

A connection failure can be determined based on whether a backend target is not responding, or based on specified HTTP status codes that should be treated as failures of a specific backend target. For example, if a backend target is responding to requests with a "500 Server Error" status, it may be desirable to consider that server down even though the web service is responsive.

If multiple targets are specified in a site configuration but a load balancing strategy is not applied, the Availability Profile will cause the first listed target in the site configuration to be used unless it fails. Secondary targets will only be used if the first target is not available.

Currently, the only availability profile type is **On-Demand**. You may wish to create different profiles for different sites based on differing site needs for retry counts, retry delays, timeouts, or HTTP status codes.

Creating an availability profile

You can create an availability profile to define when a backend server is considered failed.

Steps

1. Go to **Networking# Availability Profiles**, then click **Add Availability Profile**.
2. Enter a unique descriptive name for the profile.
3. Select the **On-Demand** Type.
4. Enter the number of milliseconds to wait for a connection to be established to a backend target in the **Connect Timeout (ms)** field.
5. Enter the number in milliseconds the amount of time to wait before timing out the request for a pooled connection to the target site in the **Pooled Connection Timeout (ms)** field. Enter -1 for no timeout.
6. Enter the number in milliseconds the amount of time to wait before timing out the read of the response from a target site in the **Read Timeout (ms)** field. Enter -1 for no timeout.
7. Enter the number of times to retry a connection to a backend target before considering the target failed in the **Max Retries** field.
8. Enter the number of milliseconds to wait between retries in the **Retry Delay (ms)** field.

9. Enter the number of seconds to wait before trying a failed target again in the **Failed Retry Timeout (s)** field.
10. Optionally enter a list of HTTP status codes that should be considered as a failure in the **Failure HTTP Status Codes** field. The sequence for this list is not important.
11. Click **Save**.

Editing an availability profile
You can edit the properties of an existing availability profile.

Steps

1. Navigate to **Networking**# **Availability Profiles**, expand the desired profile, then click ✎.
2. Make the required changes to the profile.
3. Click **Save**.

Deleting an availability profile
You can delete an existing availability profile.

Steps

1. Navigate to **Networking**# **Availability Profiles**.
2. Expand the desired profile, then click 🗑.
3. Click **Delete** to confirm.

HTTP Requests
The settings for HTTP Requests are used to match a served resource with the originating client when one or more reverse proxies are between the client and the served resource.

For example, when a reverse proxy sits between the client and the PingAccess server or a PingAccess agent, the additional proxy might be identified as the client. Such proxies can be configured to inject additional headers to relay the originating client address.

The Host Source and Protocol Settings allow PingAccess to determine the effective URL of a request using a list of alternative headers. PingAccess uses this URL to apply security policies and perform HTTP redirects.

When the PingAccess Agent is behind a load balancer that is performing HTTPS offload, the load balancer must inject the Host Source and Protocol Source headers.

The **IP Source** section lets you specify an ordered list of header names to identify the source IP address. By default, `X-Forwarded-For` is configured as a heading.

The **Host Source** section lets you specify an ordered list of header names to identify the host source name. The Host Source options are only valid in proxy deployments. By default, `X-Forwarded-Host` and `Host` are configured as headings.

The **Protocol Source** section can be used to define the header that identifies the protocol used for the original request. The default value is `X-Forwarded-Proto`.

Configuring an alternative IP Source header
You can configure an ordered list of header names to identify the source IP address.

Steps

1. Navigate to **Networking**# **HTTP Requests**
2. In the **IP Source** section of the page, enter a header name to search for in the **Header Names** list.
3. If desired, click 🗑 to the right of an existing header to delete it from the list.

4. Select either `First` or `Last` for the **List Value Location** to determine whether, when a list of values is in the header, the first value or the last value in the list should be used as the IP Source value. The default value is `Last`.
5. Enable or Disable the **Fallback to Last Hop IP** checkbox to determine, if none of the listed headers is present in the request, whether the upstream IP address should be used for rule evaluation. If this value is disabled and no headers match, the network range rule will return a `Forbidden` status.

   Note: This option uses the specified headers in an agent deployment, and uses networking layer information in a proxy deployment.

6. Click **Save**.

Configuring an alternative host source header
You can configure an ordered list of header names to identify the host source name.

About this task

Note: Host Source settings are valid for proxy deployments only.

Steps

1. Navigate to **Networking#** **HTTP Requests**
2. In the **Host Source** section of the page, click **Header Names** to enter a header name to search for in the **Header Names** list.
3. If desired, click 🗑 to the right of an existing header to delete it from the list.
4. Select either `First` or `Last` for the **List Value Location** to determine, when a list of values is in the header, if the first value or the last value in the list should be used as the **Host Source** value. The default value is `Last`.
5. Click **Save**.

Configuring an alternative protocol source header
You can configure a header name for a protocol source.

Steps

1. Navigate to **Networking#** **HTTP Requests**.
2. In the **Protocol Source** section of the page, enter a header name in the **Header Name** field.
3. Click **Save**.

Listeners
The Listeners configuration page is used to assign key pairs to the active HTTPS listeners, as well as to define additional listener ports for the PingAccess engine.

If you configure a trusted certificate group for a virtual host, or configure an engine key pair to associate it with a virtual host, those settings are used instead of any applicable HTTPS listeners or engine listeners for the virtual host.

Cipher Suite ordering for HTTPS Listeners

PingAccess supports the use of a defined order for cipher suite usage to help ensure the most secure cipher suites are used first, regardless of the client request. The cipher suite order is defined in `<PA_HOME>/conf/run.properties` using the `tls.default.cipherSuites` property.

On new installs, or in the case of an upgrade to release 5.1 or later, this behavior is the default. You can disable this behavior and specify PingAccess to use the order provided by the client by setting `useServerCipherSuiteOrder` to `false` via the PingAccess API `/httpsListeners` endpoint.

HTTPS Listeners
PingAccess listens for HTTPS requests on the ADMIN, ENGINE, and AGENT ports in all deployments, and on the CONFIG QUERY port in clustered deployments.

A key pair must be assigned to each listener. See *Key Pairs* for information on setting up a key pair.
By default, the listeners are configured for HTTPS and use pre-generated key pairs associated with `localhost`.

HTTPS Listener Descriptions

HTTPS Listener	**Description**
ADMIN	Listens for requests for the administrative user interface and the PingAccess REST APIs.
ENGINE	Listens for requests from PingAccess Agents.
AGENT	Listens for HTTP or HTTPS requests that are proxied to target web servers associated with *Sites*.
CONFIG QUERY	Listens for requests for configuration information from replica administrative nodes and engine nodes in clustered deployments.

Assigning a Key Pair to an HTTPS Listener
You can assign a new key pair for any of the active HTTPS listeners.

Steps

1. Navigate to **Networking**# **Listeners**.
2. In the **HTTPS Listeners** section of the page, click the drop-down menu to the right of the listener and select a key pair.
3. Click **Save**.

Engine key pairs
Assigning a Key Pair to a Virtual Host
You can assign a key pair to a virtual host.

Steps

1. Navigate to **Networking**# **Listeners**
2. In the **Engine Key Pairs** section of the page, click **Edit** in the row the desired key pair appears in
3. Use the drop-down list to select select the virtual hosts for which the key pair should be used.
4. Click **Save**.

Engine listeners
Defining an engine listener
You can define a new engine listener.

Steps

1. Navigate to **Networking**# **Listeners**
2. Click **Add Engine Listener**.
3. Enter a descriptive name for the listener.

4. Enter the port the listener will open.

> ⓘ **Info:** Remember to open the port in the system firewall, or the listener will not be able to process any incoming requests.

5. If the port should listen for HTTP connections, clear the **Secure** option.

> ⓘ **Note:** By default, engine listeners listen for HTTPS connections to protect sensitive data.

6. Select a configured trusted certificate group to use for certificate authentication.
7. Click **Save**.

Editing an engine listener
You can modify the properties of an existing engine listener.

Steps

1. Navigate to **Networking**# **Listeners**
2. In the **Engine Listeners** section, expand an existing engine listener.
3. Click ✎.
4. Make the required changes.
5. Click **Save**.

Deleting an engine listener
You can delete an existing engine listener.

Steps

1. Navigate to **Networking**# **Listeners**
2. In the **Engine Listeners** section, expand an existing engine listener.
3. Click 🗑.
4. Click **Delete** to confirm.

Load Balancing Strategies
Load Balancing Strategies are used in a Site configuration to distribute the load between multiple backend target servers. Load balancing settings are optional and only available if more than one target is listed for a site.

This functionality can replace a load balancer appliance between the PingAccess engine nodes and the target servers, allowing for a simpler network architecture.

The load balancing strategies currently available are `Header-Based` and `Round Robin`.

The `Header-Based` strategy requires a header be included in the request that defines the target to select from the **Site** configuration. This strategy has an option to fall back if the requested target is unavailable, or if the header is missing from the request.

The `Round Robin` strategy has a sticky session option that permits a browser session to be pinned to a persistent backend target. This strategy works in conjunction with the availability profile to select a target based on its availability, and the load balancer will not select a target that is in a failed state.

Configuring a Load Balancing Strategy
You can create a new load balancing strategy.

Steps

1. Go to **Networking**# **Load Balancing Strategies**
2. Click **Add Load Balancing Strategy**.

3. Enter a unique descriptive name for the strategy.
4. Select the **Type**, either `Header-Based` or `Round Robin`.
5. Configure the options for the selected Load Balancing Strategy type:
    - For a `Header-Based` Load Balancing Strategy:
        a. In the **Header Name** field, enter the name of the header that contains the selected target host.
        b. If desired, click **Show Advanced** and select the **Fall Back to First Available Host** option to tell PingAccess to use the first available target defined for the site if the target specified in the header is not available or if the header is not present in the request.

        > **Note:** If this option is not enabled and the specified target is not available or the request header is not present, the client will receive a `Service Unavailable` response.
    - For a `Round Robin` Load Balancing Strategy:
        a. If browser sessions should not be pinned to a persistent backend target, clear the **Sticky Session Enabled** option. This option is enabled by default.
        b. If the **Sticky Session Enabled** option is enabled, enter a cookie name to use in the **Cookie Name** field. This cookie is used by the PingAccess engine to track the persistent backend targets for a session.

        > **Note:** When a web session is defined, the **Cookie Name** field defines a cookie prefix to use. The rest of the cookie name comes from the **Audience** field in the Web Session.

Editing a Load Balancing Strategy
You can edit the properties of an existing load balancing strategy.

Steps

1. Go to **Networking**# **Load Balancing Strategies**.
2. Expand the load balancing strategy you want to edit, then click .
3. Make any desired changes to the profile.
4. Click **Save**.

Deleting a load balancing strategy
You can delete an existing load balancing strategy.

Steps

1. Go to **Networking**# **Load Balancing Strategies**.
2. Expand the load balancing strategy you want to edit, then click .
3. Click **Delete** to confirm.

Proxies
This page contains the forward proxy configuration used when PingAccess makes requests to Sites or Token Providers.
Adding a proxy
You can add a forward proxy configuration to be used when PingAccess makes requests to sites or token providers.

Steps

1. Navigate to **Networking**# **Proxies**.
2. Click **Add Proxy**.
3. Specify a **Name** for the proxy configuration.

4. Enter an optional **Description**.
5. Enter the **Host** name for the forward proxy.
6. Enter the **Port** number for the forward proxy.
7. If the forward proxy requires authentication, select the **Requires Authentication** checkbox.
8. If required, enter the **Username** for the forward proxy.
9. If required, enter the **Password** for the forward proxy.
10. Click **Save**.

Editing a proxy

You can edit the properties for an existing proxy.

About this task

This task allows you to edit a proxy configuration.

> **Note:** If you edit a proxy configuration that is associated with an engine or replica administrative node, you must download and install a new configuration on those nodes.

Steps

1. Navigate to **Networking**# **Proxies**.
2. Expand an existing proxy configuration.
3. Click ✎.
4. Make the required changes.
5. Click **Save**.

Deleting a proxy

You can delete an existing proxy.

Steps

1. Navigate to **Networking**# **Proxies**.
2. Expand an existing proxy configuration.
3. Click 🗑.
4. Click **Delete** to confirm.

Security

The **Security** tab contains controls for certificates and key pairs.

Choose from one of the following sections:

- *Certificates* on page 200
- *Key Pairs* on page 202

Certificates

You can import certificates into PingAccess to establish anchors used to define trust to certificates presented during secure HTTPS connections.

Outbound secure HTTPS connections such as communication with PingFederate for OAuth access token validation, identity mediation, and communication with a target Site require a certificate trusted by PingAccess. If one does not exist, communication is not allowed.

Certificates used by PingAccess may be issued by a CA or self-signed. CA-issued certificates are recommended to simplify trust establishment and minimize routine certificate management operations. Implementations of an X.509-based PKI (PKIX) typically have a set of root CAs that are trusted, and the root certificates are used to establish chains of trust to certificates presented by a client or a server during communication.

The following formats for X.509 certificates are supported:

- Base64 encoded DER (PEM)
- Binary encoded DER

A Certificate Group is a trusted set of anchor certificates used when authenticating outbound secure HTTPS connections. The Java Trust Store group contains all the certificates included in the keystore located in the Java installation at `$JAVA_HOME/lib/security/cacerts`. This group of certificates contains well-known, trusted CAs. If you are connecting to Sites that make use of certificates signed by a CA in the Java Trust Store, you do not need to create an additional Trusted Certificate Group for that CA. You cannot manage the Java Trust Store group from the PingAccess administrative console. Expand a section for steps to import and manage certificates and create and manage trusted certificate groups.

Importing a certificate
You can import a new certificate.

Steps

1. Navigate to **Security# Certificates**.
2. Click **+** to the right of the **Certificates** subheading.
3. Enter an **Alias** for the certificate.
4. Click **Choose File** to select the certificate.
5. Click **Add** to import the certificate. A new certificate row appears on the Certificates page.

> **Note:** If the Certificate is either expired or not yet valid, PingAccess displays a warning, but the import will proceed.

Deleting a certificate
You can delete an existing certificate.

Steps

1. Navigate to **Security# Certificates**.
2. Expand the certificate you want to delete.
3. Click 🗑.
4. When prompted, click **Delete** to confirm the deletion request.

> **Info:** If the certificate is associated with a trusted certificate group, you cannot delete it.

Creating a trusted certificate group
You can create a new trusted certificate group.

Steps

1. Navigate to **Security# Certificates**.
2. Click **+** to the right of the **Trusted Certificate Groups** heading.
3. Drag a certificate onto the box that appears.
4. Enter a **Name** for the group in the box that appears.
5. Select the **Use Java Trust Store** checkbox to set the new group to include the Java Trust Store group. For example, if you create your own intermediate CA certificate that is signed by a well-known CA in the Java Trust Store.

6. Select the **Skip certificate date checks** checkbox to allow PingAccess to ignore date-related errors for certificates that are not yet valid or have expired.
7. Click **Add**.
8. Additional certificates can be added to the new trusted certificate group by dragging them into the group.

Adding a certificate to a trusted certificate group
You can add a certificate to an existing trusted certificate group.

Steps

1. Navigate to **Security**# **Certificates**.
2. Drag a certificate into an existing trusted certificate group.

Editing a trusted certificate group
You can edit the properties of an existing trusted certificate group.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Expand the trusted certificate group you want to edit.
   - Add a certificate to the group by dragging it into the group from the certificate list.
   - Delete a certificate from the group by clicking **-** to the right of the certificate.
   - Edit the trusted certificate group parameters by clicking and then making your changes. If you edit these options, click **Save** to save them.

Removing a certificate from a trusted certificate group
You can remove a certificate from a trusted certificate group.

Steps

1. Expand the Trusted Certificate Group containing the certificate you want to remove.
2. Click on the certificate you want to remove.

Deleting a trusted certificate group
You can delete an existing trusted certificate group.

Steps

1. Navigate to **Security**# **Certificates**.
2. Expand the trusted certificate group you want to delete.
3. Click .
4. When prompted, click **Delete** to confirm the deletion request.

Key Pairs
PingAccess provides built-in Key Pairs, which are required for secure HTTPS communication.

A Key Pair includes a private key and an X.509 certificate. The certificate includes a public key and the metadata about the owner of the private key.

PingAccess listens for client requests on the administrative console port and on the PingAccess engine port. To enable these ports for HTTPS, the first time you start up PingAccess, it generates and assigns a Key Pair for each port. These generated Key Pairs are initially assigned on the **HTTPS Listeners** page.

Additionally, Key Pairs are used by the **Mutual TLS Site Authenticator** to authenticate PingAccess to a target Site. When initiating communication, PingAccess presents the client certificate from a Key Pair to the Site during the mutual TLS transaction. The Site must be able to trust this certificate in order for authentication to succeed.

ⓘ **Info:** Ensure that the administrative console node and engines in a cluster have the same cryptographic configuration. For example, if you generate an elliptic curve Key Pair on the administrative console and the engines in the cluster are not configured to support elliptic curve Key Pairs, then the engines are not able to use that Key Pair for the engine **HTTPS Listeners** or as the Key Pair in a **Mutual TLS Site Authenticator**. Cryptographic configuration differences are often caused by having a Java Cryptographic Extension with limited strength providers installed (see the *Oracle Java documentation* for more information).

Importing an existing key pair
You can import a key pair from a PKCS#12 file.

Steps

1. Navigate to **Security# Key Pairs**.
2. Click **Import**.
3. In the **Alias** field, enter a name that identifies the key pair. Special characters and spaces are allowed. This name identifies the key pair when assigning the key pair to various configurations such as *HTTPS Listeners*.
4. Enter the **Password** used to protect the PKCS#12 file. PingAccess uses the password to read the file.
5. Click **Choose File** to locate the PKCS#12 file.
6. Click **Save** to import the file.

ⓘ **Note:** If the key pair is either expired or not yet valid, PingAccess displays a warning, but the import will proceed.

Generating a new key pair
You can generate a key pair and self-signed certificate.

Steps

1. Navigate to **Security# Key Pairs**.
2. Click **Add Key Pair**.
3. Enter the fields required for the new key pair.

   If the key pair is going to be used for incoming requests on multiple hosts or multiple IP addresses, enter additional **Subject Alternative Names** to meet those requirements.
4. Click **Save**.

Generating a Certificate Signing Request
You can generate a certificate signing request (CSR) to establish more security and trust than using a self-signed certificate.

Steps

1. Go to **Security# Key Pairs**.
2. Click **Generate CSR** for the certificate you want to generate a CSR for. PingAccess generates a CSR file and your browser will download it.
3. Provide this file to a Certificate Authority (CA). The CA signs the file and provides a CSR Response that you can upload and use to replace the self-signed certificate. If the CA is well known, its certificates are installed by default in most browsers, and the user is not prompted to trust an unknown certificate.
4. When you receive the CSR response, follow the instructions at *Importing a Certificate Signing Request Response* on page 204.

Importing a Certificate Signing Request Response
You can import a CSR response to replace the self-signed certificate in a key pair.

About this task

Click **CSR Response** and fill out the form.

Note: Before you import the CSR Response, import the signing CA certificate into PingAccess and add it to a *Trusted Certificate Group*.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Click **CSR Response** for the key pair the CSR applies to.
3. Select the **Trusted Certificate Group** to use for validating that the certificate in the CSR Response is correctly formed.
4. Choose the CSR Response file.
5. Click **Save**.

Adding a certificate to a key pair
You can add a certificate to an existing key pair. You start with a leaf certificate, then add the intermediate and root certificates as required.

About this task

Note: To modify the certificates included in a chain, remove the certificates from the key pair and add them again or delete the certificate and recreate it by importing a new certificate file and adding certificates to the key pair.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Expand an existing key pair.
3. At the bottom of the key pair chain certificate list, click **Add Certificate**.
4. Click **Choose File** to browse for and select the certificate file.
5. Click **Add**.

Removing a certificate from a key pair
You can remove a certificate from an existing key pair.

About this task

Note: This procedure removes the last certificate in the chain. Certificates can only be removed in reverse order.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Expand an existing key pair.
3. To remove the last certificate in the chain, click .
4. Click **Delete** to confirm.

Downloading a Certificate

You can download a certificate when you need to configure a peer to trust a certificate used by PingAccess.

About this task

For example, download the certificate for the key pair used by a **Mutual TLS Site Authenticator** and configure the target site to trust the certificate.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Click **Download Certificate** in the row for the certificate you want to download.
3. Your browser will download the certificate and save it in your local filesystem.

Deleting a key pair

You can delete an existing key pair.

About this task

Info: If a key pair is currently in use, you cannot delete it.

Steps

1. Navigate to **Security**# **Key Pairs**.
2. Expand the key pair you want to delete.
3. Click 🗑.
4. When prompted, click **Delete** to confirm the deletion request.

System

The **System** tab contains controls for the administrative UI.

Choose from one of the following sections:

- *Admin Authentication* on page 205
- *Configuration Export/Import* on page 210
- *Clustering* on page 211
- *License* on page 216
- *Token Provider* on page 217
- *Token Validation* on page 225

Admin Authentication

This page controls the PingAccess administrator authentication method. The default PingAccess administrator authentication method used to protect the administrative console is basic authentication (username and password).

You can use one of these authentication methods:

- Basic Authentication – Use a single set of credentials with the username `Administrator` and a password provided in the admin UI.
- Single Sign-On (SSO) – Use an OpenID Connect Token Provider to provide authentication.

For either authentication method, you can configure session properties such as the session cookie type and timeout values.

You can configure the authentication methods used for accessing the administrative APIs.

Configuring basic authentication

You can configure basic authentication for the administrative user interface.

About this task

The authentication default for the PingAccess administrative console is HTTP Basic Authentication. Basic Authentication uses the HTTP Authorization header to transmit the username and password credentials. The PingAccess server response contains a `PA_UI` cookie, which is a signed JSON Web Token. Subsequent HTTP requests send this cookie for authentication rather than the less secure HTTP Authorization header.

Basic Authentication supports one user – Administrator. This username cannot be changed. If you want to allow more than one user to access the admin UI, you should use SSO authentication.

Steps

1. Go to **System**# **Admin Authentication**# **Basic Authentication**.
2. Click **Enable**.
3. Click **Save**.
4. Go to **System**# **Admin Authentication**# **Admin UI**.

   Note: The current authentication setting is included in the menu title. For example, if basic authentication is configured as it is by default, the menu option is **Admin UI – Basic**.

5. In the **Authentication Method** section, select **Basic Authentication**.
6. Click **Save**.

Changing the password for basic authentication

You can change the password used for basic authentication.

Steps

1. Go to **System**# **Admin Authentication**# **Basic Authentication**.
2. Enter the current administrator password.
3. Enter and confirm the new password.

   Important: The new password must meet the configured password complexity rules defined in `pa.admin.user.password.regex` in `run.properties`.

4. Click **Save**.

Configuring Admin UI SSO Authentication

You can configure single sign-on for the administrative user interface.

Before you begin

Prerequisites:

There are several configuration steps required within the OIDC token provider as well as PingAccess that you must complete to enable SSO. Expand a section to view those configurations. The Administrative SSO option can be configured to require a specific authentication mechanism, leveraging the OIDC token provider Requested AuthN Context Selector using the PingAccess *authentication requirements* options.

- The OIDC provider configuration must be completed. See the configuration instructions for your OIDC token provider:
  - *Configure PingFederate runtime*
  - *Configure PingOne*
  - *Configure OpenID connect*
- The OIDC token provider server certificate must be *imported* into a trusted certificate group, and that trusted certificate group must be associated with the OIDC token provider runtime.
- If you are using PingFederate, you must have a **profile** scope set up in PingFederate that includes the *openid*, *profile*, *address*, *email*, and *phone* scope values (see PingFederate documentation for *Configuring a Client*).

If you are using PingFederate as the OIDC token provider, when you configure the client in PingFederate, use the following options:

- The **Client Authentication** must be set to `None`
- The **Allowed Grant Types** must be set to `Implicit`
- The **Redirect URIs** must include `https://<PA_Admin_Host>:<PA_Admin_Port>/*`
- If you are not using administrative roles in PingAccess, the OpenID Connect **Policy** should be set to a policy that uses issuance criteria to restrict access based on some additional criteria.

  Warning: If the selected OpenID Connect Policy does not use issuance criteria to limit which users can be granted an access token, *ALL* users in the associated identity store configured in PingFederate will be able to authenticate to the PingAccess Admin console and make changes. See *Identifying Issuance Criteria for Policy Mapping* in the *PingFederate Administrator's Manual*.
- If you are configuring administrative roles to enable the PingAccess auditor role, the issuance criteria defined in PingFederate should be defined to allow either an administrator or an auditor to be issued an access token. The attribute contract defined in the OpenID Connect Policy must include the additional attribute data that will be used to define the user's role in PingAccess.

About this task

Use the Single Sign-On (SSO) Authentication page in PingAccess to enter the Client ID for the OAuth Client you created in the OIDC token provider.

Info: Complete the configuration for connecting to the PingFederate OAuth AS on the *Configuring PingFederate for PingAccess SSO* on page 221 page as well as completing the steps below.

Steps

1. Go to **System# Admin Authentication# Admin UI**.

   Note: The current authentication setting is included in the menu title. For example, if basic authentication is configured as it is by default, the menu option is **Admin UI – Basic**.
2. In the **Authentication Method** section, select **Single Sign-On**.

   Tip: To define a fall back administrator authentication method if the OIDC token provider is unreachable, enable the `admin.auth=native` property in `run.properties`. This overrides any configured administrative authentication to *Basic Authentication*.

3. In the **OpenID Connect Login Type** dropdown, select a login type:
    - Code – The standard OIDC login flow. This option is the default.
    - POST – A login flow using the form_post response mode, which returns response parameters as application/x-www-form-urlencoded HTML form values.
    - x_post – A login flow based on OIDC that passes claims from the provider via the browser using the implicit grant type.
4. In the **Client ID** field, enter the unique identifier assigned when you created the PingAccess OAuth client within your OIDC token provider.
5. In the **Client Secret** field, if you selected the Code login type or if you have enabled session validation, enter the secret assigned when you created the OAuth relying party client within your OIDC token provider.
6. Optional: If your environment requires an authentication requirements list, in the **Authentication Requirements** dropdown, select a defined athentication requirements list, or click **Create** to create a new list.
7. Optional: If you want to enable advanced settings, click **Show Advanced** and use one or more of the advanced options:
    a. Optional: To request one or more scopes from the OIDC token provider, select one or more scopes in the **Scopes** dropdown. If you have a token provider configured, published scopes are available to select from the list based on the selected Client ID. You can specify unverified scopes by typing the scope and clicking **Use unverified scope "[scopename]"**.

       Your token provider must be properly configured to handle all of the requested scopes you specify, including any custom scope values.

       Note: The user can access all attributes by examining browser traces. While they are integrity protected to prevent changes, any sensitive or confidential attributes can be viewed should the user decode the ID Token's value.

    b. Optional: To validate sessions with the configured PF instance during request processing, in the **Validate Session** options, click **Yes**. This option is not supported by PingOne or third-party OIDC token providers.
    c. Optional: To periodically refresh user data from the OIDC token provider, in the **Refresh User Attributes** options, click **Yes** and specify a **Refresh User Attributes Interval** in seconds.
    d. Optional: If you want PingAccess to cache user attribute information for use in policy decisions, click **Cache User Attributes**. When this option is disabled, user attribute information is encoded and stored in the session cookie.
    e. Optional: To enable the use of single logout, click **Use Single-Logout**. This option must be configured in the OIDC provider.

       Note: If you are using PingFederate as a token provider, you should enable the **Check For Valid Authentication Session** in the PingFederate access token management configuration to prevent session replay.

8. Optional: If you want to enable role-based authorization, click **Enable Roles** and configure one or more roles:
    a. In the Administrator section, click **Add Required Attribute** once or more.

       For a role to be granted, all configured attribute values must match.
    b. Enter an **Attribute Name** and **Attribute Value** for each required attribute.

       Note: If you are using PingFederate as a token provider, the attribute name is defined in PingFederate under **OAuth Settings**# **OpenID Connect Policy Management**# *Your_Policy* #

Attribute Contact as an extension to the contract. The value to use depends on the configuration of the **Contract Fulfillment** tab for the policy.

The attribute named `group` in your attribute contract may be mapped to an LDAP server attribute source that contains a `groupMembership` attribute. A valid group membership for the administrator might be the group `cn=pingaccess-admins,o=myorg`. In this example, you would use `group` as the **Attribute Name** and `cn=pingaccess-admins,o=myorg` as the **Attribute Value**.

c. Optional: If you want to add auditors, click **Add Required Attribute** in the Auditor section once or more.
d. Optional: Enter an **Attribute Name** and **Attribute Value** for each required attribute.

9. Click **Save**.

Configuring session properties

You can configure session properties for the administrative user interface.

Steps

1. Go to **System# Admin Authentication# Admin UI**.

Note: The current authentication setting is included in the menu title. For example, if basic authentication is configured as it is by default, the menu option is **Admin UI – Basic**.

2. Specify the **Cookie Type**, Encrypted JWT or Signed JWT.
3. Specify the unique **Audience** name the token is applicable to.
4. Specify an **Idle Timeout** in minutes. This sets the length of time you want the PA Token to remain active when no activity is detected. When the idle expiration is reached, the session automatically terminates.
5. Specify a **Max Timeout** in minutes. This sets the length of time you want the PA Token to remain active. Once the PA Token expires, an authenticated user must re-authenticate.
6. Specify an **Expiration Warning** in minutes. This specifies the point at which a user will be warned of upcoming session expiry.
7. Specify the **Session Poll Interval** in seconds. This sets the length of time between user info poll requests for the admin UI.
8. Click **Save**.

Configuring API Authentication

You can configure authentication for the administrative API.

About this task

Info: For more information on the PingAccess Administrative API, see *Administrative API Endpoints*.

Steps

1. Go to **System# Admin Authentication# Admin API OAuth**.

Note: The current API authentication setting is included in the menu title. For example, if Admin API OAuth is disabled, the menu option is **Admin API OAuth – Disabled**.

2. Select **Enable** to enable API OAuth authentication.
3. Enter the **Client ID** assigned when you created the OAuth client for validating OAuth access tokens (for more information about configuring a client ID in PingFederate, see *Configuring a Client*).
4. Enter the **Client Secret** assigned when you created the OAuth client in the OIDC token provider.

5. Select the **Scope** required to successfully access the API. For more information about defining scopes in PingFederate, see *Authorization Server Settings*.
6. Enter the **Subject Attribute Name** you want to use from the OAuth access token as the subject for auditing purposes. At runtime, the attribute's value is used as the Subject field in audit log entries for the Admin API.
7. If you are using administrator/auditor roles, perform the following steps:
   a. Select **Enable Roles** to enable role based authentication.
   b. Click **Add Required Attribute** to define a new attribute that is required for Administrator access.

   > **Note:** More than one attribute may be defined here; if more than one is defined, all attribute values must match in order to grant access for the role.

   c. Enter the **Attribute Name** returned in the access token and the **Attribute Value** that defines the user as an administrator.

   > **Note:** The attribute name used here is defined in PingFederate under **OAuth Settings# OpenID Connect Policy Management#** *Your_Policy* # **Attribute Contact** as an extension to the contract. The value to use depends on the configuration of the **Contract Fulfillment** tab for the policy.

   The attribute named `group` in your attribute contract may be mapped to an LDAP server attribute source that contains a `groupMembership` attribute. A valid group membership for the administrator might be the group `cn=pingaccess-admins,o=myorg`. In this example, you would use `group` as the **Attribute Name** and `cn=pingaccess-admins,o=myorg` as the **Attribute Value**.
   d. If you want to define criteria for an auditor, select **Enable Auditor Role**.
   e. Define criteria for the auditor in the same way you did for the administrator role.
   f. Click **Add Required Attribute** to add an additional attribute.
8. Click **Save** to activate API Authentication.

Configuration Export/Import
The Configuration Export/Import options create and restore a full PingAccess configuration.

You can back up and restore your configuration for disaster recovery or for testing purposes. You can restore your configuration on the same system or a new system, as long as the version used on the restore system matches the version used when the backup was made.

The configuration backup is stored as a `json` file, and contains the entire PingAccess configuration, with the exceptions of the administrative user configuration and the keys used for JWTs. It uses the same format as results from the Administrative APIs.

> **CAUTION:** As the exported `json` file contains much of your PingAccess configuration, ensure the file is stored somewhere with appropriate security controls in place.

Sensitive data such as secrets and passwords are encrypted in the backup file. If you have *configured a master key encryptor* using the Addon SDK for Java, the host key file (JWK Set) is encrypted and included in the exported data.

You can modify the backup file directly. If you plan to do so, make an unmodified copy of the backup file before you begin.

Exporting a PingAccess configuration
You can export your current configuration.

About this task

Large PingAccess configurations can take up to half an hour to export. During an export, the PingAccess configuration cannot be modified.

Steps

1. Navigate to **System**# **Configuration Export/Import**.
2. Click the **Download** button under **Export Configuration**. The downloaded file name is `pa-data-<timestamp>.json`.

> ⓘ **Note:** The *<timestamp>* value is formatted `MM-DD-YYYY.hh.mm.ss` - so a date and time of January 31, 2015 1:35 PM would be encoded as `01-31-2015.13.35.00` in the filename.

Importing PingAccess configuration
You can import a previously saved configuration.

About this task

The **Import Configuration** option is a version-specific tool used to import a previously exported configuration. PingAccess checks the exported `json` file to ensure that the file came from the same version of PingAccess that it is being imported into.

Large PingAccess configurations can take up to a few hours to import. During an import, the PingAccess configuration cannot be modified or read.

To import a PingAccess configuration:

Steps

1. Navigate to **System**# **Configuration Export/Import**.
2. Under the **Import Configuration** heading, click **Choose File**.
3. Select the json export file containing the configuration to import.
4. Click **Import** to start the import process.
5. When prompted for confirmation, click **Confirm**.

> ⓘ **Important:** This operation is destructive, and overwrites your *entire* PingAccess configuration. Passwords in the system will revert to what they were when the backup was created. Unless you perform a backup prior to restoring a different configuration, the configuration prior to clicking **OK** will not be recoverable.

6. If the Agent or Admin listener key pairs change as a result of the import operation, restart PingAccess.
7. If the environment is clustered, ensure that the engines are using the proper engine keys. If they are not, re-save the engine to generate a new public key, and reconfigure the engine to use the newly generated key.

Clustering
PingAccess can be configured in a clustered environment to provide higher scalability and availability for critical services.

While it is important to understand that there may be tradeoffs between availability and performance, PingAccess is designed to operate efficiently in a clustered environment.

PingAccess clusters are made up of three types of nodes:

Administrative Console

Provides the administrator with a configuration interface

Replica Administrative Console

Provides the administrator with the ability to recover a failed administrative console using a manual failover procedure.

Clustered Engine

Handles incoming client requests and evaluates policy decisions based on the configuration replicated from the administrative console

Any number of clustered engines can be configured in a cluster, but only one administrative console and one replica administrative console can be configured in a cluster.

Configuration information is replicated to all of the clustered engine nodes and the replica administrative node from the administrative console. State information replication is not part of a default cluster configuration, but some state information can be replicated using PingAccess subclusters.

PingAccess Subclusters

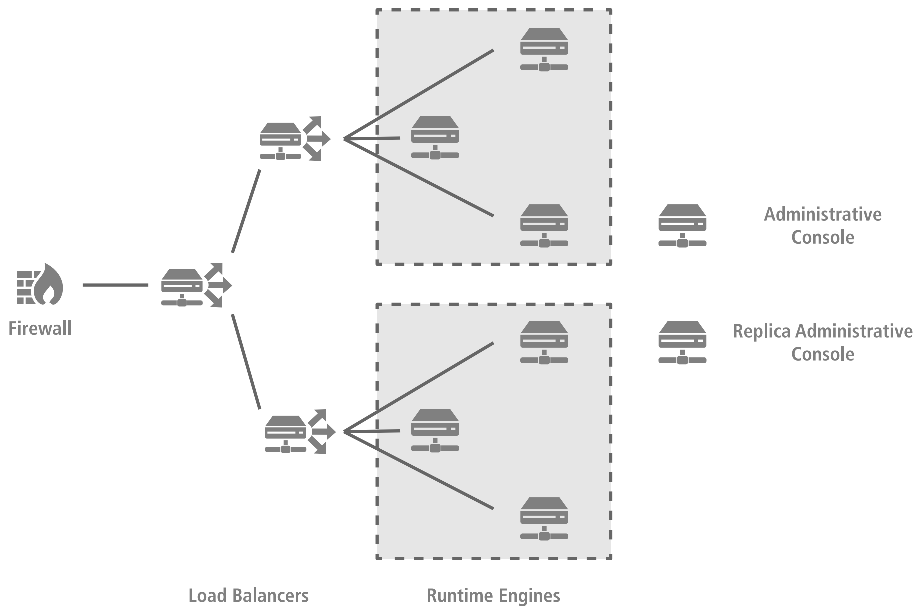
Subclusters are a method to provide better scaling of very large PingAccess deployments by allowing multiple engine nodes in the configuration to share certain information. A load balancer is placed in front of each subcluster in order to distribute connections to the nodes in the subcluster.

Subclusters serve three purposes:

- Providing fault-tolerance for mediated tokens if a cluster node is taken offline.
- Reducing the number of STS transactions with PingFederate when the front-end load balancer does not provide a sticky session.
- Ensure rate limits are enforced properly if the front-end load balancer does not provide a sticky session.

If token mediation and rate limiting are not used in your environment, subclustering is not necessary.

Info: This cache can be tuned using the EHCache Configuration Properties listed in the **Configuration Properties** documentation.



PingAccess provides clustering features that allow a group of PingAccess servers to appear as a single system. When deployed appropriately, server clustering can facilitate high availability of critical services. Clustering can also increase performance and overall system throughput. It is important to understand, however, that availability and performance are often at opposite ends of the deployment spectrum. Thus, you may need to make some configuration tradeoffs that balance availability with performance to accommodate specific deployment goals.

In a cluster, you can configure each PingAccess engine, or node, as an administrative console, a replica administrative console, or a runtime engine in the `run.properties` file. Runtime engines service client requests, while the console server administers policy and configuration for the entire cluster (via the administrative console). The replica administrative console provides a backup copy of the information on the administrative node in the event of a non-recoverable failure of the administrative console node. A cluster may contain one or more runtime nodes, but only one console node and only one replica console node. Server-specific configuration data is stored in the PingAccess administrative console server in the run.properties file. Information needed to bootstrap an engine is stored in the **bootstrap.properties** file on each engine.

At startup, a PingAccess engine node in a cluster checks its local configuration and then makes a call to the administrative console to check for changes. How often each engine in a cluster checks the console for changes is configurable in the engine run.properties file.

Configuration information is replicated to all engine nodes. By default, engines do not share runtime state. For increased performance, you can configure engines to share runtime state by configuring cluster interprocess communication using the run.properties file.

Info: Runtime state clustering consists solely of a shared cache of security tokens acquired from the PingFederate STS for **Token Mediation** use cases using the **Token Mediator Site Authenticator**.

Engine nodes include a status indicator that indicates the health of the node and a **Last Updated** field that indicates the date and time of the last update. The status indicator can be green (good status), yellow (degraded status), or red (failed status).

The status is determined by using the value for `admin.polling.delay` as an interval to measure health:

Green (good status):

The replica administrative node contacted the primary administrative node on the last pull request.

Yellow (degraded status):

The replica administrative node contacted the primary administrative node between 2 and 10 intervals.

Red (failed status):

The replica administrative node has either never contacted the primary administrative node, or it has been more than 10 intervals since the nodes communicated.

Engines
Configuring an engine node
You can configure an engine node as part of a cluster.

About this task
For each engine:

Steps

1. Click **System# Clustering**
2. Click **+ Add Engine** to configure a new engine.
3. Enter a **Name** for the engine. Special characters and spaces are allowed.
4. Enter a **Description** of the engine.
5. If applicable, specify an **HTTP Proxy** for the engine. Click **Create** to create an HTTP proxy.
6. If applicable, specify an **HTTPS Proxy** for the engine. Click **Create** to create an HTTPS proxy.
7. Specify the **Engine Trusted Certificate** to use for cases where a TLS-terminating network appliance, such as a load balancer, is placed between the engines and the admin node.
8. Click **Save & Download** to generate and download a public and private key pair into the *<enginename>*`_data.zip` file for the engine. This file is prepended with the name you give the engine. Depending on your browser configuration, you may be prompted to save the file.

9. Copy the zip file to the *PA_HOME* directory of the corresponding engine in the cluster and unzip it. The engine uses these files to authenticate and communicate with the administrative console.

   ⓘ **Info:** Generate a new key for an engine at any time by clicking **Save & Download** and unzipping the `<enginename>_data.zip`

10. On Linux systems running the PingAccess engine, change the permissions on the extracted archive on the engine to replace the files with a new set of configuration files. When that engine starts up and begins using the new files, PingAccess deletes the old key.`pa.jwk` to mode 400 by executing the command `chmod 400 conf/pa.jwk` after extracting the zip file.
11. Start each engine.

   ⓘ **Info:** For information on configuring engine to share information with each other in a cluster, see *Configure a PingAccess Cluster*.

Editing an engine node
You can edit the name and description of an engine node within your cluster, and download a new public key if necessary.

Steps

1. Go to **System#** **Clustering**.
2. Expand the node you want to edit, then click ✎.
3. Edit the node **Name** or **Description**, as appropriate.
4. If a new public key is needed, click **Save & Download**.
5. If a new public key is not needed, click **Save**.

Removing an engine node
You can remove an engine node from the cluster.

Steps

1. Go to **System#** **Clustering**.
2. Expand the engine node you want to delete and click 🗑 to permanently remove all references to the node from the cluster.
3. Click **Delete** in the confirmation window.

Administrative nodes
Configuring the administrative node
You can configure one PingAccess node as the administrative node.

About this task

This procedure allows you to specify an HTTP or HTTPS proxy. If proxy configuration is defined in a properties file (`bootstrap.properties` or `run.properties`), it will take precedence over UI or API configuration.

If a proxy is configured on a replica administrative node, when failing over and before removing the `bootstrap.properties` file, the administrative node should have the same proxy configuration.

ⓘ **Warning:** If you are promoting a replica administrative node to an administrative node, remove the bootstrap properties file from the replica administrative node.

Steps

1. Go to **System#** **Clustering#** **Administrative Nodes**.

2. Enter the host and port for the administrative console. The default is `localhost:9000`.
3. If applicable, specify an **HTTP Proxy** for the engine. Click **Create** to create an HTTP proxy.
4. If applicable, specify an **HTTPS Proxy** for the engine. Click **Create** to create an HTTPS proxy.
5. Click **Save**.

Configuring the replica administrative node
You can configure one PingAccess node as a replica administrative node to provide an alternative if the administrative node fails.

About this task

When using a replica administrative node, you must define a key pair to use for the CONFIG QUERY listener that includes both the administrative node and the replica administrative node. You can do this either by using a wildcard certificate or by defining subject alternative names in the key pair that include the replica administrative node's DNS name. If you use a replica administrative node in your configuration, configure the replica administrative node before defining the engine nodes, or the `bootstrap.properties` files generated for the engine nodes will not include information about the replica administrative node.

In addition to the configuration below, the Replica Administrative node includes a status indicator that indicates the health of the node and a read-only **Last Updated** field that indicates the date and time of the last update. The status indicator can be green (good status), yellow (degraded status), or red (failed status).

The status is determined by using the value for `admin.polling.delay` as an interval to measure health:

Green (good status):

The replica administrative node contacted the primary administrative node on the last pull request.

Yellow (degraded status):

The replica administrative node contacted the primary administrative node between 2 and 10 intervals.

Red (failed status):

The replica administrative node has either never contacted the primary administrative node, or it has been more than 10 intervals since the nodes communicated.

Note: If you are configuring a replica administrative node in the environment, that must be done before you configure the engines.

Steps

1. Go to **System**# **Clustering**# **Administrative Nodes**.
2. Configure the **Host** value. This name and port pair must match either a subject alternative name in the key pair or be considered a match for the wildcard specified if the key pair uses a wildcard in the common name.
3. If applicable, specify an **HTTP Proxy** for the engine. Click **Create** to create an HTTP proxy.
4. If applicable, specify an **HTTPS Proxy** for the engine. Click **Create** to create an HTTPS proxy.
5. Specify the **Replica Administrative Node Trusted Certificate** to use for cases where a TLS-terminating network appliance, such as a load balancer, is placed between the engines and the admin node.
6. Click **Save & Download** to download the *`<replicaname>`*`_data.zip` file for the replica administrative node. PingAccess automatically generates and downloads a public and private key pair into the `bootstrap.properties` file for the node. The **Public Key** is indicated on this screen.
7. Copy the downloaded file to the replica administrative node's *`PA_HOME`* directory and unzip it.

8. If the replica administrative node is running on a Linux host, execute the command **`chmod 400 conf/pa.jwk`**.
9. Edit *`PA_HOME`*`/conf/run.properties` on the replica administrative node and change the `pa.operational.mode` value to `CLUSTERED_CONSOLE_REPLICA` .
10. Start the replica administrative node.
11. Verify replication has completed by monitoring the *`PA_HOME`*`/log/pingaccess.log` file and looking for the message "Configuration successfully synchronized with administrative node".

License

This page lets you view the details of your PingAccess license or upload a new license.

The **System**# **License** page displays the details of the current license.

You can upload a new license file using this page. The new license is compared to the existing license, and unusual discrepancies such as the new license having a sooner expiration date than the existing license are flagged. After reviewing any warnings, you can choose to import the new license file, replacing the existing file.

In a clustered environment, the license file on the administrative node is replicated to all of the engine nodes and the replica administrative node. The engine nodes do not require a license to function, but some default templates appear differently depending on the information in the license.

When a license is about to expire or has expired, the UI displays a warning, and a WARNING-level message is added to the PingAccess Console log.

Note: If the installation has a running configuration, and the admin shuts down the server, removes the license file from the filesystem, and restarts the server, the existing runtime configuration will continue to work. However, the admin will have to install a new license file on the filesystem, or upload one via the UI, to enable accessing and applying changes via the UI.

Uploading a PingAccess license

You can upload a new PingAccess license file.

About this task

When you select a new license file to import, PingAccess compares the new license file's attributes with those of the current license before installing it. The UI displays a warning ribbon on the page in certain cases. For example, if the expiration date of the uploaded license is sooner than the current license, the UI will flag that as a warning.

If the new license is acceptable, you can commit it, and the new license will replace the old.

Steps

1. Navigate to **System**# **License** to view the attributes of the installed license.
2. Scroll down to the **Import License** section.
3. Click **Choose a File** to select a license file.

Warning: The UI will display a warning ribbon for the following cases:

- **Expiration date:**

  The new license is set to expire on a date earlier than that of your current license.
- **Expired:**

  The new license is already expired.
- **License version:**

The major version of the license doesn't match the current version of PingAccess.

- **Max Applications:**

  The new license is limited to support fewer applications than your current license.

4. Click **Import** to import the selected license.

   ⓘ **Note:** If you selected the wrong license, you can either click **Remove** to remove the selected license from the **Import License** section of the page, or you can click **Choose File** to select a different license file.

5. Click **Confirm** to install the selected license.

Token Provider

You can configure the token provider for PingAccess. The supported token providers are PingFederate, PingOne, and common providers using the OpenID Connect protocol.

PingFederate

- *Configuring PingFederate runtime* on page 217
- *Configuring PingFederate administration* on page 220
- *Configuring the OAuth Resource Server* on page 220
- *Configuring PingFederate for PingAccess SSO* on page 221

PingOne

- *Configuring PingOne* on page 223

Common Token Provider

- *Configuring OpenID Connect* on page 223
- *Configuring OAuth Authorization Server* on page 225

PingFederate

Configuring PingFederate runtime

You can configure a secure connection to the PingFederate runtime.

Before you begin

Before configuring a secure connection to the PingFederate runtime, it is necessary to export the PingFederate certificate and import it into a trusted certificate group in PingAccess. Perform the following steps:

1. In PingFederate, export the certificate active for the Runtime Server. See *SSL Server Certificates* in the *PingFederate Administrator's Manual* for more information.
2. Import the certificate into PingAccess.
3. Create a Trusted Certificate Group if one does not already exist.
4. Add the certificate to a Trusted Certificate Group.

About this task

ⓘ **Info:** For information on setting PingFederate up as an OAuth Authorization Server, see *Enabling the OAuth AS* and *Authorization Server Settings* in the PingFederate documentation.

Once the PingFederate Runtime connection is saved, PingAccess will test the connection to PingFederate. If the connection cannot be made, a warning will be displayed in the admin interface, and the PingFederate runtime is prevented from saving.

The steps displayed depend on your environment. In a new deployment, some of the PingFederate configuration information is imported automatically from the PingFederate well-known endpoint. If you have upgraded from PingAccess 5.2 or earlier and have an existing token provider configuration, this information is provided manually. If you have performed an upgrade and want to see the new version of this page, configure the token provider using the `/pingfederate/runtime` API endpoint. For more information, see *Administrative API Endpoints*.

> **Note:** Configuring PingFederate as a token provider using the `/pingfederate/runtime` overwrites the existing PingFederate configuration.

Once you have successfully configured the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **View Metadata# Refresh Metadata**.

Configure the connection to the PingFederate Runtime

Steps

1. Go to **System# Token Provider**.
2. Click **PingFederate**.
3. Click the **Runtime** tab.
4. In the **Issuer** field, enter the PingFederate issuer name.
5. Optional: Enter a **Description** for the PingFederate instance.
6. From the **Trusted Certificate Group** list, select the certificate group the PingFederate certificate is in. This list is available only if you select **Secure**.
7. To configure advanced settings, click **Show Advanced**.
8. If hostname verification for secure connections is not required for either the runtime or the back channel servers, select the **Skip Hostname Verification** check box.
9. Optional: To use a configured proxy for back channel requests, select the **Use Proxy** check box.

   > **Note:** If the node is not configured with a proxy, requests are made directly to PingFederate.
10. Optional: Select **Use Single-Logout** to enable single logout. To use this feature, SLO must be configured on the OIDC provider.
11. Optional: Enter the **STS Token Exchange Endpoint** to be used for token mediation if it is different from the default value of `<issuer>/pf/sts.wst`.
12. Click **Save**.

    > **Info:** After you save this configuration and *Configuring the OAuth Resource Server* on page 220, a PingFederate Access Validator is available for selection when you define OAuth-type rules in Policy Manager.

Configure PingFederate runtime (original workflow)

If your PingAccess deployment has been upgraded from version 5.2 or earlier with an existing token provider configuration, and you have not configured a token provider using the `/pingfederate/runtime` API endpoint, this workflow is displayed for configuring a PingFederate runtime.

Before you begin

Before configuring a secure connection to the PingFederate runtime, it is necessary to export the PingFederate certificate and import it into a trusted certificate group in PingAccess. Perform the following steps:

1. In PingFederate, export the certificate active for the Runtime Server. See *SSL Server Certificates* in the *PingFederate Administrator's Manual* for more information.
2. Import the certificate into PingAccess.

3. Create a Trusted Certificate Group if one does not already exist.
4. Add the certificate to a Trusted Certificate Group.

About this task

ⓘ **Info:** For information on setting PingFederate up as an OAuth Authorization Server, see *Enabling the OAuth AS* and *Authorization Server Settings* in the PingFederate documentation.

Once the PingFederate Runtime connection is saved, PingAccess will test the connection to PingFederate. If the connection cannot be made, a warning will be displayed in the admin interface, and the PingFederate runtime is prevented from saving.

Once you have configured the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **View Metadata# Refresh Metadata.**

Steps

1. Go to **System# Token Provider**.
2. Click the **PingFederate** radio option.
3. Click the **Runtime** tab.
4. In the **Host** field, enter the PingFederate runtime host name or IP address for the PingFederate Runtime.
5. In the **Port** field, enter the PingFederate runtime port number.
6. Optional: In the Base Path field, enter the base path, if needed, for the PingFederate runtime. It must start with a slash - for example: `/federation`.
7. Click the **Audit Level** check box to log information about the transaction to the audit store. PingAccess audit logs record a selected subset of transaction log information at runtime and are located in the `/logs` directory of your PingAccess installation.
8. Select the **Secure** check box if PingFederate is expecting HTTPS connections.
9. From the **Trusted Certificate Group** list, select the certificate group the PingFederate certificate is in. This field is available only if you select **Secure** in the previous step.
10. To configure advanced settings, click **Show Advanced**.
11. Click **Add Back Channel Server**.
12. Enter one or more *`hostname:port`* pairs in the **Back Channel Servers** list.
13. If the back channel uses HTTPS, enable the **Back Channel Secure** option. This option becomes available when at least one back channel server is defined.
14. If the back channel uses an alternate base path, enter the path in the **Back Chanel Base Path** field.
15. If hostname verification for secure connections is not required for either the Runtime or the Back Channel Servers, enable the **Skip Hostname Verification** option.
16. If hostname verification is required, enter the host name PingAccess should expect in the **Expected Hostname** field.
17. To use a configured proxy for back channel requests, select the **Use Proxy** check box.

    ⓘ **Note:** If the node is not configured with a proxy, requests are made directly to PingFederate.

18. Select **Use Single-Logout** to enable single logout. To use this feature, SLO must be configured on the OIDC provider.

19. Click **Save**.

> ⓘ **Info:** After you save this configuration and *Configuring the OAuth Resource Server* on page 220, a PingFederate Access Validator is available for selection when you define OAuth-type rules in Policy Manager.

Configuring PingFederate administration
You can configure your PingFederate administration settings.

About this task

For information on the PingFederate Administration API see *PingFederate Administrative API* in the PingFederate documentation.

Once the PingFederate Administration configuration is saved, PingAccess will test the connection to PingFederate. If the connection cannot be made, an error will be displayed in the admin interface, and the configuration will not be saved.

Steps

1. Navigate to **System**# **Token Provider**# **Administration**.
2. Enter the **Host** name or IP address for access to the PingFederate Administrative API.
3. Enter the **Port** number for access to the PingFederate Administrative API.
4. Optional: Enter the **Base Path** for the PingFederate Administrative API.

   The **Base Path** must start with a slash (/).

   For example: `/path`
5. Enter the **Admin Username**.

   This username only requires Auditor (read only) permissions in PingFederate.
6. Enter the **Admin Password**.
7. Select **Audit Level** to log information about the transaction to the audit store. PingAccess audit logs record a selected subset of transaction log information at runtime and are located in the `/logs` directory of your PingAccess installation.
8. Enable **Secure** if PingFederate is expecting HTTPS connections.
9. From the **Trusted Certificate Group** list, select the group of certificates to use when authenticating to PingFederate. PingAccess requires that the certificate in use by PingFederate anchor to a certificate in the associated Trusted Certificate Group. This field is available only if you enable **Secure**.
10. To configure advanced settings, click **Show Advanced**.
11. To use a configured proxy for API requests, select the **Use Proxy** checkbox.

> ⓘ **Note:** If the node is not configured with a proxy, requests are made directly to PingFederate.

12. Click **Save**.

> ⓘ **Tip:** To view OIDC metadata provided by the token provider, click **View Metadata** after saving the token provider configuration.

Configuring the OAuth Resource Server
When receiving OAuth-protected API calls, PingAccess acts as an OAuth Resource Server, checking with the PingFederate OAuth Authorization Server on the validity of the bearer access token it receives from a client. In order to validate the token, a valid OAuth client must exist within the PingFederate OAuth Authorization Server.

Before you begin
Prior to configuring this option, PingFederate Administration Configuration must be completed.

About this task

ⓘ **Note:** This configuration is optional and needed only if you plan to validate PingFederate OAuth access tokens.

Steps

1. Navigate to **System# Token Provider# OAuth Resource Server**
2. Enter the OAuth **Client ID** you defined when creating the PingAccess OAuth client in PingFederate.

   ⓘ **Info:** When you configure an OAuth client in PingFederate, be sure to select *Access Token Validation* as the allowed grant type. For more information, see *Configuring a Client* in the *PingFederate Administrator's Manual*.
3. Enter the **Client Secret** you defined when creating the PingAccess OAuth client within PingFederate.
4. Select **Cache Tokens** to retain token details for subsequent requests. This option reduces the communication between PingAccess and PingFederate.
5. If **Cache Tokens** is enabled, specify the **Token Time To Live** by entering the number of seconds to cache the access token. The default value of -1 means no limit. This value can be -1 or above and must be less than the PingFederate Token Lifetime.
6. In the **Subject Attribute Name** field, enter the attribute you want to use from the OAuth access token as the subject for auditing purposes. For example, `username`. At runtime, the attribute's value is used as the Subject field in audit log entries for API Resources with policies that validate access tokens. The attribute must align with an attribute in the *OAuth access token attribute contract* defined within PingFederate.
7. If multiple Access Token Managers are configured in PingFederate, select the **Send Audience** option to send the URI the user requested as the `aud` OAuth parameter to select an Access Token Manager.

   ⓘ **Note:** Use of this option requires that the Access Token Management instances be configured with appropriate Resource URIs. Matching of the Resource URI is performed on a most-specific match basis.
8. To enable the use of OAuth 2.0 token introspection select the **Use Token Introspection Endpoint** option.

   ⓘ **Note:** This option is only supported with PingFederate 8.2 or later.
9. Click **Save** to save your changes.

Configuring PingFederate for PingAccess SSO
You can configure PingFederate to enable administrator SSO for PingAccess.

About this task

To enable administrator SSO to PingAccess, configure the following settings within the PingFederate AS. Click the icon ( ) next to each section heading to access additional configuration information. For example, click next to **Roles and Protocols** to open a new window and view the Choosing Roles and Protocols page of the PingFederate documentation.

ⓘ **Note:** The information below is an example configuration and does not cover all required steps for each PingFederate OAuth Settings page discussed, only fields necessary for successful SSO to the

PingAccess administrative console. Fields not mentioned are not necessary for this configuration (see *Using OAuth Menu Selections* for configuration details of the PingFederate OAuth Settings pages).

ⓘ **Note:** You must complete the configuration for connecting to the PingFederate OAuth AS instance you plan to use (see *Configuring PingFederate administration* on page 220).

Roles and Protocols

- Enable the OAuth 2.0 AS role and the OpenID Connect protocol.
- Enable the IdP Provider role and a protocol.

Password Credential Validator (PCV)

- Create a PCV for authenticating administrative users.

Adapters

- Create an HTML Form IdP Adapter and specify the PCV you configured.

Authorization Server Settings

- Select **Implicit** in the Reuse Existing Persistent Access Grants for Grant Types section.

Access Token Management

- Select **Internally Managed Reference Tokens** as the Access Token Management Type.
- Extend the contract by adding the Username attribute on the Access Token Attribute Contract page.

OpenID Connect Policy Management

ⓘ **Info:** We recommend creating an OpenID Connect Policy to use specifically for PingAccess administrative console authentication.

- Delete all of the attributes that appear in the Extend the Contract section of the Attribute Contract page. The only required attribute is **sub**.
- Select **Access Token** as the Source and **Username** as the Value on the Contract Fulfillment page.

Client Management

ⓘ **Info:** We recommend creating a Client to use specifically for PingAccess administrative console authentication.

- Select **None** for Client Authentication.
- Add the location of the PingAccess host as a Redirect URI. For example: `https://localhost:9000/*`
- Select **Implicit** as an Allowed Grant Type.
- Select one of the elliptic curve (**ECDSA**) algorithms as the OpenID Connect ID Token Signing Algorithm and select the OpenID Connect Policy to use for PingAccess administrative console authentication.

IdP Adapter Mapping

- Map the HTML Form IdP Adapter Username value to the USER_KEY and the USER_NAME contract attributes for the persistent grant and the user's display name on the authorization page, respectively.

Access Token Mapping

- Map values into the token attribute contract by selecting **Persistent Grant** as the Source and **USER_KEY** as the value for the Username attribute. These are the attributes included or referenced in the access token.

PingOne

Configuring PingOne

You can configure PingOne as the token provider.

Before you begin

You must have the PingOne issuer ID, or have access to the PingOne console, to perform this procedure.

About this task

Once you have successfully configured the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **View Metadata**# **Refresh Metadata**.

Steps

1. Go to **System**# **Token Provider**.
2. Select **PingOne**.
3. In the **Issuer** field, enter the PingOne issuer ID. This information is available in the PingOne console.
4. Optional: Enter a description for the connection.
5. In the **Trusted Certificate Group** list, select a trusted certificate group that PingAccess will use when authenticating to PingOne.
6. To configure the connection to use a configured proxy, click **Show Advanced** and select **Use Proxy**.
7. Click **Save**.

Common

Configuring OpenID Connect

You can configure OpenID Connect token provider settings.

About this task

Once you have successfully configured the token provider, click **View Metadata** to display the metadata provided by the token provider. To update the metadata, click **View Metadata**# **Refresh Metadata**.

Steps

1. Go to **System**# **Token Provider**.
2. Click **Change Token Provider Type**.
3. When prompted to confirm the change, click **Confirm**.
4. Select **OpenID Connect**.
5. In the **Issuer** field, enter the OpenID Connect provider's issuer identifier.
6. In the **Description** field, enter a description for the token provider.
7. Select the **Audit Level** check box to record requests to OpenID Connect provider to the audit store.
8. From the **Trusted Certificate Group** list, select the group of certificates to use when authenticating to OpenID Connect provider. PingAccess requires that the certificate in use by OpenID Connect provider anchor to a certificate in the associated Trusted Certificate Group.
9. If required, click **Add Query Parameter** and enter custom query parameter name and value pairs used by the OpenID Connect provider.
10. To configure advanced settings, click **Show Advanced**.

11. To use a configured proxy, select the **Use Proxy** check box.

    > **Note:** If the node is not configured with a proxy, requests are made directly to the token provider.

12. Select **Use Single-Logout** to enable single logout. To use this feature, SLO must be configured on the OIDC provider.
13. Click **Save**.

Configure token provider specific options

You can configure plugins that perform particular functions for the selected token provider type.

In the case of the PingAccess for Azure AD solution, the plugin addresses the following problems:

- **Transform data** - The format of data returned from the OIDC UserInfo endpoint results in some unexpected JSON formatting. This data is transformed into a format that PingAccess can easily digest.
- **Use the Azure AD Graph API** - If the groups attribute contains more than 200 groups, the id_token contains a level of indirection that points to a URL in the Azure AD Graph API. Through the creation of a simple purpose-driven application, you can communicate with the Azure ID Graph API to retrieve the complete list of groups.
- **Retrieve group display names** - The groups attribute is a list of GUIDs. The groups for a user are only provided as GUIDs since User-friendly names for AAD groups are not globally unique. Configure the Graph API call to include the group names along with the GUID for creation of more robust policies.

1. First, *Create the Azure AD Graph API application*
2. Next, *Configure token provider specific options*

Prerequisite: Create the Azure AD Graph API application

To use the Azure AD Graph API, an application must exist whose purpose is to provide an **Application ID** and **Key** that PingAccess will use as the **Client ID** and **Client Secret** for communication with the Graph API. Create the application in Azure AD via the **App Registrations** blade using the following criteria:

- **Name** - Enter a unique name for the application, for example: *Graph API app*
- **Application Type** - Web app / API
- **Sign-on URL** - This field is not relevant for this particular use case, but is required by Azure AD. Enter the PingAccess host.

1. When the application has been created, navigate to the application in the list.
2. Select **Required permissions** and click **Add**.
3. Choose the **Windows Azure Active Directory** API, select the following permission, and click **Save**.
    - Application Permissions - Read directory data
4. Copy the **Application ID**.
5. Generate and copy a **Key**.

Configure token provider specific options

1. Navigate to **System# Token Provider**. Scroll to the **Token Provider Specific Options** section.
2. From the **Type** list, select **Azure Active Directory**.
3. Select the **Use Azure AD Graph API** checkbox.
4. In the **Client ID** field, enter the **Application ID** you copied from the Azure AD API application you created.
5. In the **Client Secret** field, paste the **Key** you copied.
6. Select **Retrieve Group Display Names**.

> **Important:** To retrieve groups data for a particular application in the token, the manifest for that application must be modified to include a Group Membership Claim. In the **App Registrations** blade,

select the application and click the **Manifest** button. Locate the `"groupMembershipClaims"` entry and specify a group type, for example `"SecurityGroup"`.

7. Click **Save**.

Configuring OAuth Authorization Server
You can configure the OAuth Authorization Server.

Steps

1. Navigate to **System**# **Token Provider**.
2. Select **OAuth Authorization Server**.
3. In the **Description** field, enter a description for the authorization server.
4. In the **Targets** field, enter one or more `hostname:port` pairs for the OAuth Authorization Server. Click **Add Target** to add additional targets.
5. In the **Introspection Endpoints** field, specify the OAuth endpoint through which the token introspection operation is accomplished.
6. Optional: Select the **Audit Level** checkbox to record requests to OAuth Authorization server to the audit store.
7. Select the **Secure** checkbox if OAuth Authorization server is expecting HTTPS connections. When selected, use the **Trusted Certificate Group** list to select the group of certificates to use when authenticating to OAuth Authorization Server. PingAccess requires that the certificate in use by OAuth Authorization Server anchor to a certificate in the associated Trusted Certificate Group.
8. In the **Client ID** field, enter the unique identifier assigned when you created the PingAccess OAuth client within your OAuth Authorization Server.
9. In the **Client Secret** field, enter the Client Secret associated with the Client ID.
10. Optional: Select the **Cache Tokens** to retain token details for subsequent requests. This option reduces the communication between PingAccess and OAuth Authorization Server. When selected, use the **Token Time To Live** checkbox to enter the number of seconds to cache the access token. A value of -1 means there is no limit. This value should be less than the OAuth Authorization Server Token Lifetime.
11. In the **Subject Attribute Name** field, enter the attribute you want to use from the OAuth access token as the subject for auditing purposes. At runtime, the attribute's value is used as the Subject field in audit log entries for API Resources with policies that validate access tokens.
12. Select the **Send Audience** checkbox to send the URI the user requested as the `aud` OAuth parameter for PingAccess to the OAuth 2.0 Authorization Server.
13. To configure advanced settings, click **Show Advanced**.
14. To use a configured proxy, select the **Use Proxy** checkbox.
15. Click **Save**.

Note: If the node is not configured with a proxy, requests are made directly to the token provider.

Token Validation
API and Web + API applications can be configured to use access token validators to locally verify signed and/or encrypted access tokens. This feature works in conjunction with token providers that support JWS and/or JWE validation.

Tip: When using PingFederate as the token provider for this feature, export the `Generated: ENGINE` keypair located at **Security**# **Key Pairs** from PingAccess and import to PingFederate Trusted CAs.

Adding an access token validator
You can add an access token validator to verify signed or encrypted access tokens.

Steps

1. Go to **System#** **Token Validation**.
2. Click **Add Access Token Validator**.
3. Specify a **Name** for the token validator.
4. In the **Type** list, select the type of key you want to validate. The type of key is specified in token provider configuration.

> **Note:** For information related to PingFederate configuration, see *Configure JSON token management* for information about the configuration of PingFederate.

5. Enter a **Description** for the token validator.
6. In the **Path** field, specify the endpoint path used to verify the signature. This entry must start with a slash (/), and must not end with a slash (/). Host and port are derived from PingFederate token provider configuration. A query string is permitted in the path.
7. In the **Subject Attribute Name** field, enter the attribute expected as the subject. If the specified SAN is not present in the token, validation will fail. This is a required field.
8. In the **Issuer** field, enter the expected value of the issuer that is to be included in the access token. If configured, and the value is not present in the token, validation will fail.
9. In the **Audience** field, specify the audience value that is to be included in the access token. If configured, and the value is not present in the token, validation will fail.
10. Click **Save**.

Editing an access token validator
You can edit an existing access token validator.

Steps

1. Go to **System#** **Token Validation**.
2. Expand the application on the **Properties** tab and click ✎.
3. Make the required changes.
4. Click **Save**.

Deleting an access token validator
You can delete an existing access token validator.

Steps

1. Go to **System#** **Token Validation**.
2. Expand the access token validator you want to delete and click 🗑.
3. Click **Delete** to confirm.

Configure OAuth key management
You can configure settings for OAuth key management.

Steps

1. Go to **System#** **Token Validation**.
2. Select the **Key Roll Enabled** checkbox to enable key rolling. Deselect to disable key rolling.
3. Specify the interval at which you want to roll keys by entering a value, in hours, in the **Key Roll Enabled** field.
4. Click **Save**.

Agents and Integrations

PingAccess for AWS

PingFederate Environment Requirements for PingAccess for AWS

Introduction

In addition to deploying a demonstration environment that includes a basic PingFederate instance, you can configure the PingAccess for AWS solution to deploy with support for an existing PingFederate environment. You may choose this installation method for test or production deployments with a PingFederate environment that is external to your AWS VPC.

This document is a guide for administrators that describes the components the automation process is expecting to be configured in your PingFederate environment, how to configure those components, and how to include information about your environment to the automation.

While variations to these PingFederate requirements do exist, and while it is possible to deviate from these instructions in some cases, this document is intended to describe a minimum configuration and, as such, these instructions would be outside the scope of this document at this time.

At a minimum, your PingFederate environment requires the following components:

1. *Install the AWS Password Credential Validator*
2. *Configure the AWS PCV instance*
3. *Configure a Simple Password Credential Validator Instance*
4. *Create an IdP adapter*
5. *Add a Persistent Grant Extended Attribute*
6. *Configure default scope values*
7. *Configure an Access Token Manager instance*
8. *Configure an AWS PCV Resource Owner Credentials Mapping*
9. *Configure an IdP adapter mapping*
10. *Configure an IdP Adapter Access Token mapping*
11. *Configure an AWS PCV Access Token mapping*
12. *Create an OpenID Connect policy*
13. *Configure the PingAccess Admin SSO OAuth client*
14. *Configure the Resource Owner API Client OAuth client*
15. *Create and (optionally) export a PingFederate certificate*

Create an IAM User in AWS

This document describes the steps required to create an IAM User that will be used by the AWS Password Credential Validator.

Note: If your PingFederate instance resides in Amazon EC2, do not use these steps. The preferred method is instead to create an IAM Role and assign the proper permissions:

1. Navigate to **Services#** **IAM**.
2. Click **Roles**.
3. Click **Create role**.
4. Select **AWS service#** **EC2** and specify the **EC2** use case.
5. Click **Next: Permissions**.
6. Filter for **IAMReadOnlyAccess** and select this item.

7. Click Next: Review.
8. Provide a **Role name** and optional **Role description**.
9. Click **Create role**.
10. Assign this role to your PingFederate EC2 instance by right clicking the EC2 instance and choosing **Instance Settings# Attach/Replace IAM Role**.

1. In AWS, navigate to **Services# Security, Identity & Compliance# IAM**.
2. Click **Users**.
3. Click **Add User**.
4. Specify a **User name**. For example: `awspcv_api`
5. For **Access** type, select **Programmatic access**.
6. Click **Next**.
7. Select **Attach existing policies directly**.
8. Filter the list for **IAMReadOnlyAccess** and select this item.
9. Click **Next**.
10. Click **Create User**.
11. Copy the **Access key ID** for use with the AWS PCV instance that you will create in PingFederate.
12. In the **Secret access key** field, click **Show**.
13. Copy the **Secret access key** for use with the AWS PCV you will create.

Important: You must copy the Secret access key at this stage. The key will be masked permanently when you navigate away from this page.

14. Click **Close**.

PingFederate configuration requirements

The following sections describe the PingFederate configuration elements expected by the automation. At a minimum, your PingFederate environment should include these components.

Install the AWS Password Credential Validator

The following steps describe how to install the AWS password credential validator (PCV).

1. Obtain the PCV file, `aws-password-credential-validator-<version>.jar`.
2. Copy the file to your PingFederate installation at `%PingFedHome%/pingfederate/server/default/deploy`.
3. Restart PingFederate.

Configure the AWS PCV instance

The following steps describe how to configure the AWS PCV instance.

1. In PingFederate, navigate to **Server Configuration# Password Credential Validators**.
2. Click **Create New Instance**.
3. Specify the **Instance Name**: AWS PCV
4. Specify the **Instance ID**: AwsPcv
5. Select the **Instance Type**: AWS Username Password Credential Validator
6. Click **Next** .

7. Enter the **Access Key** and **Secret Key** for the *IAM user* you created.

> ⓘ **Note:** If your PingFederate instance is in EC2, and you have *created an IAM Role*, leave these fields blank.

> ⓘ **Note:** If your PingFederate instance is located in the same VPC, you can leave these fields blank.

8. Click **Next**.
9. Click **Done**.
10. Click **Save**.

Configure a Simple Password Credential Validator Instance

The following steps describe how to configure the Simple PCV instance. This PCV will be assigned to an IdP adapter instance.

> ⓘ **Note:** You can use an existing PCV/Adapter configuration for Admin SSO if it meets these configuration requirements.

1. In PingFederate, navigate to **Server Configuration**# **Password Credential Validators**.
2. Click **Create New Instance**.
3. Specify the **Instance Name**: SimplePcv
4. Specify the **Instance ID**: SimplePcv
5. Select the **Instance Type**: Simple Username Password Credential Validator
6. Click **Next**.
7. Click **Add a new row to 'Users'**.
8. Specify a **Username** and **Password** to use for PingAccess Admin SSO, confirm the password, and click **Update**. For example: joe/2Federate.
9. Click **Next**.
10. Click **Done**.
11. Click **Save**.

Create an IdP adapter instance

The following steps describe how to create an IdP adapter instance.

1. Navigate to **IdP Configuration**# **Adapters**.
2. Click **Create New Instance**.
3. Specify the **Instance Name**: HtmlFormIdpAdapter
4. Specify the **Instance ID**: HtmlFormIdpAdapter
5. Select the **Instance Type**: HTML Form IdP Adapter
6. Click **Next**.
7. Click **Add a new row to 'Credential Validators'**.
8. Select the **SimplePcv** PCV you created and click **Update**.
9. Click **Next** (x2).
10. On the **Adapter Attributes** screen, select the **Pseudonym** checkbox.
11. Click **Next** (x2).
12. Click **Done**.
13. Click **Save**.

Add a Persistent Grant Extended Attribute

The following steps describe how to add a Persistent Grant Extended Attribute.

1. Navigate to **OAuth Settings# Authorization Server Settings**.
2. In the **Persistent Grant Extended Attributes** section, type `ACTIONS` and click **Add**.
3. Scroll to the bottom and click **Save**.

Configure default scope values

The following steps describe how to add values to the default scope.

1. Navigate to **OAuth Settings# Scope Management**.
2. Specify a **Default Scope Description**: `Default Permissions`
3. Enter the following **Scope Value** and **Scope Description** items and click **Add** with each:

Scope Value	**Scope Description**
address	Address access
email	Email access
offline_access	Offline access
openid	OpenID Connect login
phone	Phone Number access
profile	Profile access

4. Click **Save**.

Configure an Access Token Manager instance

The following steps describe how to create an Access Token Manager instance.

1. Navigate to **OAuth Settings# Access Token Management**.
2. Click **Create New Instance**.
3. Enter the **Instance Name**: Internally Managed Reference Tokens
4. Enter the **Instance ID**: default
5. Select the **Instance Type**: Internally Managed Reference Tokens
6. Click **Next** (x3).
7. On the **Access Token Attribute Contract** screen, extend the contract by adding attributes **Actions** and **Username**, clicking **Add** after each.
8. Click **Next** (x3).
9. Click **Done**.
10. Click **Save**.

Configure an AWS PCV Resource Owner Credentials Mapping

The following steps describe how to create an AWS PCV Resource Owner Credentials Mapping.

1. Navigate to **OAuth Settings# Resource Owner Credentials Mapping**.
2. Select the AWS PCV from the **Source Password Validator Instance** list and click **Add Mapping**.
3. Click **Next**.
4. On the Contract Fulfillment screen, select the Source and Value for each Contract item as follows:

Contract	**Source**	**Value**
ACTIONS	Password Credential Validator	actions
USER_KEY	Password Credential Validator	rolename

5. Click **Next** (x2).
6. Click **Done**.

7. Click **Save**.

Configure an IdP adapter mapping

The following steps describe how to configure an IdP adapter mapping.

1. Navigate to **OAuth Settings**# **IdP Adapter Mapping**.
2. In the **Source Adapter Instance** list, select the **HtmlFormIdpAdapter** you created.
3. Click **Add Mapping**.
4. Click **Next.**
5. On the **Contract Fulfillment** screen, configure entries as follows:

Contract	**Source**	**Value**
ACTIONS	Text	sso
USER_KEY	Adapter	username
USER_NAME	Adapter	username

6. Click **Next** (x2).
7. Click **Save**.

Configure an IdP Adapter Access Token mapping

The following steps describe how to configure an IdP Adapter Access Token mapping.

1. Navigate to **OAuth Settings**# **Access Token Mapping**.
2. Select the `IdP Adapter: HtmlFormAdapter` from the **Context** list, `Internally Managed Reference Tokens` from the **Access Token Manager** list, and click **Add Mapping**.
3. On the **Contract Fulfillment** screen, configure entries as follows:

Contract	**Source**	**Value**
Actions	Persistent Grant	ACTIONS
Username	Persistent Grant	USER_KEY

4. Click **Next** (x2).
5. Click **Save**.

Configure an AWS PCV Access Token mapping

The following steps describe how to configure an AWS PCV Access Token mapping.

1. Navigate to **OAuth Settings**# **Access Token Mapping**.
2. Select the `Validator: AWS PCV` from the **Context** list, `Internally Managed Reference Tokens` from the **Access Token Manager** list, and click **Add Mapping**.
3. On the **Contract Fulfillment** screen, configure entries as follows:

Contract	**Source**	**Value**
Actions	Persistent Grant	ACTIONS
Username	Persistent Grant	USER_KEY

4. Click **Next** (x2).
5. Click **Save**.

Create an OpenID Connect policy

The following steps describe how to create an OpenID Connect policy.

1. Navigate to **OAuth Settings**# **OpenID Connect Policy Management**.
2. Click **Add Policy**.
3. Enter the **Policy ID**: PA-WAM
4. Enter the **Policy Name**: PA-WAM
5. Select the **Access Token Manager**: Internally Managed Reference Tokens
6. Click **Next**.
7. Delete all items beneath **Extend the Contract**.
8. Click **Next** (x2).
9. For the **Attribute Contract** `sub`, select a **Source** of `Access Token`, and a **Value** of `Username`.
10. Click **Next** (x2).
11. Click **Done**.
12. Click **Set as Default**.
13. Click **Save**.

Configure the PingAccess Admin SSO OAuth client

The following steps describe how to create the PingAccess Admin SSO OAuth client that is used by the automation.

1. Navigate to **OAuth Settings**# **Clients** and click **Create New**.
2. Enter the **Client ID**: pa_sso
3. Enter the **Name**: PingAccess Admin SSO
4. For **Client Authentication**, select **None**.
5. Enter the following **Redirect URIs** and click **Add** after each:
   - https://*:9000
   - https://*:9000/*
6. Select the following **Allowed Grant Types**:
   - Authorization Code
   - Refresh Token
   - Implicit
7. Click **Save**.

Configure the Resource Owner API Client OAuth client

The following steps describe how to create the Resource Owner API Client OAuth client that is used by the automation.

1. Navigate to **OAuth Settings**# **Clients** and click **Create New**.
2. Enter the **Client ID**: api_ro
3. Enter the **Name**: Resource Owner API Client
4. For **Client Authentication**, select **Client Secret** and specify a Secret of `2Access`.
5. Select the following **Allowed Grant Types**:
   - Resource Owner Password Credentials
   - Access Token Validation (Client is a Resource Server)
6. Click **Save**.

Create and (optionally) export a PingFederate certificate

Your PingFederate installation must use a valid certificate that contains the PingFederate hostname as either the Common Name or Subject Alternative Name. You can import a valid certificate into PingFederate or you can create a certificate using PingFederate.

The following steps describe how to use PingFederate to create and export a certificate for use with the automation.

1. Navigate to **Server Configuration# SSL Server Certificates**.
2. Click **Create New**.
3. Enter a **Common Name** that matches the URL for the PingFederate server. For example, if your PingFederate server is hosted at `mypingfed.myserver.com:9031`, the Common Name will be `mypingfed.myserver.com`.
4. Enter appropriate values for **Organization** and **Country**.
5. Click **Next**.
6. Choose to make the certificate active for both the runtime server and the admin console.
7. Click **Done**.
8. On the **Certificate Management** screen, locate the certificate you created and click **Export**.
9. Select **Certificate Only** and click **Next**.
10. On the **Export & Summary** screen, click **Export** to download the certificate.
11. Click **Done**.
12. Locate the certificate file and open in a text editor.
13. Copy and save the certificate data between the first and last lines. Do not copy the first and last lines that identify the certificate file. You will add this data to the *api.json file* used by the automation.

Modify api.json to include details about your environment

Users who wish to deploy the PingAccess for AWS solution with an existing PingFederate instance must edit the `api.json` file that is included with the automation package and supply this file to the automation. You will provide details about your PingFederate environment that will be used to configure the PingAccess token provider and Admin Authentication components.

The following instructions assume that you have followed and meet the *PingFederate configuration requirements* on page 228.

Important: It is important to note that if your AWS deployment fails for any reason while you are using a modified `api.json` file, you will need to replace the file in the S3 repository with the modified file. This is due to changes that are made to the file by the automation, such as the obfuscation of password data.

The `api.json` file is located in the automation package at `<automation package>/s3-bucket-root/pingaccess/conf`.

If you meet the PingFederate configuration requirements, the following changes will need to be made to the api.json file:

- `"pingFederate":"host":` - Enter the hostname for your PingFederate server. Do not include ports or prefixes.
- `"pingFederate":"trustedCertificateGroupName":` - Enter `"Trust Any"`.
- `"pingFederateAdmin":"host":` - Enter the hostname for your PingFederate server. Do not include ports or prefixes.
- `"pingFederateAdmin":"trustedCertificateGroupName":` - Enter `"Trust Any"` for testing or debugging purposes, otherwise *Add a PingFederate Certificate object to the automation*. Using a PingFederate certificate object overrides this property.

If your PingFederate server is running on non-default ports, enter the appropriate ports in this file.

(Optional) Add a PingFederate Certificate object to the automation

You can use an *existing PingFederate certificate* with the automation by adding the following top level object to api.json:

```
"pingFederateCertificate": {
  "alias": "PingFederate",
   "fileData": "<certificate_data>"
}
```

Replace `<certificate_data>` with the data you copied and saved in *Create and (optionally) export a PingFederate certificate* on page 232.

Add the api.json file to the automation

After you have made the appropriate changes to the `api.json` file, upload the file to your S3 bucket at `<S3 bucket>/pingaccess/conf`.

Solution Setup Guide for PingAccess for AWS

Introduction

This document provides the steps required to perform a variety of tasks related to setting up and configuring the PingAccess for AWS solution. You can choose to deploy this solution in a demonstration environment that includes a basic PingFederate instance or you can choose to deploy with an *existing PingFederate instance*.

Review the *prerequisites* prior to setting up this solution.

After you have deployed the solution, you can *configure an application*.

Setup and deployment

In this section, you will perform the tasks required to setup and deploy the PingAccess for AWS solution. Along with PingAccess, this solution includes the optional deployment and configuration of a basic PingFederate instance.

When setup is complete, you will be able to configure a protected application using PingFederate as the token provider.

To setup and deploy the solution, you will:

1. Review the setup *AWS prerequisites* on page 234
2. *Obtain the automation ZIP file* on page 236
3. *Create the PingAccess software repository* on page 237
4. *Populate the S3 bucket* on page 237
5. *Create the PingAccess key pair* on page 238
6. *Create the PingAccess VPC* on page 238
7. *Configure Security Groups* on page 243
8. *Access the PingAccess administration console* on page 244
9. *Access the PingFederate administration console* on page 245

AWS prerequisites

Prior to beginning the setup of this solution, you must first satisfy some prerequisites. These prerequisites will ensure that you have details and resources that are required for use in the setup phase. You may have one or more of these items available as part of an existing deployment. Create the items you require.

Prerequisites include the following items:

1. *Create an Amazon Web Services account* on page 235
2. *Choose your region* on page 235
3. *Choose your AMIs*
4. *Register domain name with Route53*
5. *Create a certificate*

Create an Amazon Web Services account

To make use of this solution, you need an *Amazon Web Services* account. Create an account if necessary.

> ⓘ **Important:** While many of the components required for this solution are *Free Tier* eligible, following the steps in this document may lead to incurred costs.

Choose your region

Specify the region in which your AWS environment should reside.

> ⓘ **Note:** For release 1.0, if your region does not support Signature Version 2, *modify templates to support Signature Version 4*. This action is not required for release 1.0.1 and up.

Choose your AMIs

Throughout the setup process, you will be asked to specify the ID of the Amazon Machine Image (AMI) that you want to use. AMIs are region-specific, so you will require an AMI that is available to your region.

> ⓘ **Note:** For release 1.0, if your region does not support Signature Version 2, *modify templates to support Signature Version 4*. This action is not required for release 1.0.1 and up.

> ⓘ **Important:** AMIs available from AWS have default configurations that may not meet the security requirements of some network environments. Administrators are encouraged to assess the default configurations of the AMIs and make necessary modifications to satisfy the security requirements of their environment as needed.

At this time, you may only use the AMI types specified in this document. Your configuration may include the use of only a single Amazon Linux AMI ID for all required AMI IDs, but you may choose a combination where possible using the following supported types:

> ⓘ **Note:** **Amazon Linux 2** has not been tested with this solution, and is unsupported at this time. When choosing an Amazon Linux AMI, select the **Amazon Linux AMI**.

- **Bastion host AMI ID**: Amazon Linux
- **Admin AMI ID**: Amazon Linux or RHEL 7.4
- **Engine AMI ID**: Amazon Linux or RHEL 7.4
- **PingFederate AMI ID**: Amazon Linux
- **Internal CA AMI ID**: Amazon Linux

The EC2 instance type is specified by the templates you will use. For optimal performance, use the instance type specified.

To find the ID of the AMI you want to use:

1. In AWS, navigate to **Services# Compute# EC2**.
2. On the EC2 Dashboard, click Launch Instance. This page displays the AMIs available in your selected region.
3. Make note of the AMIs you want to use, along with their AMI ID. In the following example, the AMI ID is ***ami-9fa343e7*** :

   Red Hat Enterprise Linux 7.4 (HVM), SSD Volume Type - ami-9fa343e7

> ⓘ **Note:** In release 1.0, if you are customizing the AMIs used by the setup templates, you must modify `/pingaccess/templates/security/internal-ca.yaml` and `/pingaccess/templates/`

`security/pingfederate-setup.yaml` to specify the AMI ID you want to use. By default, the specified AMI is `ami-bf4193c7`. If the specified AMI is not available to your region, setup will fail.

For more information on AMIs, see *AWS AMI documentation*.

Register a domain name with Route53

During configuration, you will be asked to specify a **Public hosted zone ID**. The public hosted zone ID maps to a domain registered with **AWS Route 53**. The domain will be used to form URLs needed for external access to a variety of components in this solution, including applications, and the PingAccess and PingFederate administration consoles.

You can purchase a domain name using Route 53, transfer an existing domain to this service, or *migrate the DNS service for a domain*.

Important:

The domain you choose to use for PingAccess admin (**Admin public host name**) and engine nodes (**Applications public host name**), as well as the demonstration PingFederate instance (**PingFederate DNS Name**), **must** be registered with and managed by AWS Route 53. Failure to register or transfer a domain to AWS Route 53 will result in setup failing.

Customers who use an existing PingFederate instance do not need to register the associated domain with AWS Route 53 if:

1. The PingFederate domain is routable via the internet.
2. PingFederate uses a different domain than will be used for PingAccess in AWS.

If an existing PingFederate instance and your PingAccess instances in AWS will use the same root domain, that domain **must** be registered with AWS Route 53.

To learn more about AWS Route 53, including how to *register a domain* or transfer an existing domain, see *AWS Route 53 documentation*.

Tip: When your domain name is available in Route 53, make note of the associated **Hosted Zone ID** so you can use it during configuration.

Create a certificate

During configuration, you will be asked to specify a Certificate ID. The certificate ID will map to a certificate issued for the domain you will use, as specified by the Public hosted zone ID.

You can request a certificate or you can import a certificate using the **AWS Certificate Manager**.

To learn more about AWS Certificate Manager, including how to request or import a certificate, see *AWS Certificate Manager documentation*.

Tip: When your certificate is available in Certificate Manager, make note of the associated **ARN** for the certificate so you can use it during configuration.

Obtain the automation ZIP file

The first step in creating your PingAccess environment in AWS is to obtain the ZIP file that includes templates and scripts used by the automation process.

Contact your Ping Identity sales representative to obtain the automation ZIP file.

Create the PingAccess software repository

In order for the automation to run successfully, you must first create a stack that will be referenced by the setup scripts. You will submit a template to *CloudFormation* that creates the S3 bucket, and also creates policies that define access to the bucket that are referenced by provisioning functions.

> ⓘ **Important:** Do not manually create the S3 bucket. Use the templates described in this document.

Prerequisite: This procedure assumes you have downloaded and unzipped the automation ZIP file.

To create the PingAccess S3 bucket stack and required policies:

1. In the AWS console, navigate to **Services**# **Management Tools**# **CloudFormation**.
2. In the CloudFormation console, click **Create Stack**.
3. On the Select Template screen, click to select **Upload a template to Amazon S3**.
4. Click **Choose File**.
5. Navigate to the root of the unzipped archive, select the `create-repo.yaml` template, and click **Open**.
6. Click **Next**.
7. On the **Specify Details** screen, specify the **Stack Name**. For example, *pingaccess-s3-stack* .
8. Specify the **Bucket Name**. For example, *pingaccess-s3-bucket* .

   > ⓘ **Important:** Bucket names must be unique ***across all of AWS*** . If your chosen bucket name is in use, the template will fail.
9. Click **Next**.
10. On the **Options** screen, click **Next.**
11. On the Review screen, click to select the **I acknowledge that AWS CloudFormation might create IAM resources** check box.
12. Click **Create**.
13. When stack creation is complete, view your S3 bucket at **Services**# **Storage**# **S3**.

Populate the S3 bucket

After you create the PingAccess software bucket in S3, you will populate your bucket with the files needed for the automation process. The automation ZIP file you downloaded contains the required directory structure. You will add files to this directory structure and upload a set of folders to S3.

> ⓘ **Note:** Do not modify the directory structure. Modifying the directory structure will cause the setup process to fail.

> ⓘ **Note:** If you are planning to redeploy with an existing configuration, see *Redeploying with an existing configuration* on page 247.

Add files to the directory structure

Prior to uploading to S3, add files that are required for setup. These files include:

- The **Java Server JRE** - Copy the Server JRE install package to `s3-bucket-root/jre/server-jre-<version>-linux-x64.tar.gz`.
- The **PingAccess distribution package** - Copy the PingAccess 5.0.0 or higher distribution package to `s3-bucket-root/pingaccess/dist/pingaccess-<version>.zip`.
- The **PingAccess license file** - Copy the PingAccess license file to `s3-bucket-root/pingaccess/licenses/pingaccess.lic`.
- The **PingFederate distribution package** (Optional) - Copy the PingFederate 8.4.1 or higher distribution package to `s3-bucket-root/pingfederate/dist/pingfederate-<version>.zip` if you are deploying the basic PingFederate instance for a demonstration environment.

- The **PingFederate license file** (Optional) - Copy the PingFederate license file to `s3-bucket-root/pingfederate/licenses/pingfederate.lic` if you are deploying the basic PingFederate instance for a demonstration environment.
- Additional **library JAR files** (Optional) - Copy any additional library JAR files (e.g. plugins) that should be included with the installation to `s3-bucket-root/automation-artifacts/<export-prefix>/pingaccess/lib`, where `<export-prefix>` is the **Export name prefix** you will supply to the *automation template*.

Upload the directory structure to S3

1. In the AWS console, navigate to **Services# Storage# S3**.
2. Select the bucket you created.
3. Click **Upload**.
4. In a file window, select the contents of the `s3-bucket-root` folder. The contents include the following folders and their contents:
   - `automation-artifacts`
   - `instance-init-util`
   - `internal-ca`
   - `jre`
   - `pingaccess`
   - `pingfederate`
5. Drag and drop the folders into the AWS **Upload** window.
6. Click **Next** (x3).
7. On the final screen, click **Upload**.

Create the PingAccess key pair

This solution will require the use of a key pair for SSH access. You can import an existing key pair, use a key pair that already exists in your AWS environment, or you can create new key pair.

Create a new key pair

1. In AWS, navigate to **Services# Compute# EC2**.
2. In the left menu bar, under the **Network & Security** heading, click **Key Pairs**.

   Tip: To import an existing key pair, select **Import Key Pair** on the Key Pairs page.
3. Click **Create Key Pair**.
4. Type the **Key pair name** and click **Create**.

   The private key is automatically downloaded to your system, while the public key is stored in AWS.

   Important: Save the private key to a safe location.

Create the PingAccess VPC

This document provides the steps required to create VPC environment. The automation script includes functions to create all of the required components for this deployment, including:

- PingAccess Administration instance
- PingAccess Engine instances
- Load balancers
- Routing configuration
- Security groups configuration
- PingFederate instance (Optional)

To deploy into an existing VPC, see *Deploy PingAccess into an existing VPC* on page 239.

Important: If you are planning to deploy the PingAccess for AWS solution along with an existing PingFederate instance, your PingFederate configuration must meet the components requirements expected by the automation. To review the PingFederate configuration requirements, see *PingAccess for AWS: PingFederate environment requirements*.

Note: If you are redeploying with an existing configuration, see *Redeploying with an existing configuration* on page 247.

To create the complete VPC environment:

1. In AWS, navigate to your S3 bucket.
2. Navigate to `/pingaccess/templates` and click on the `create-pingaccess.yaml` template.
3. On the **Overview** tab, copy the template link.
4. Navigate to **Services# Management Tools# CloudFormation**.
5. Click **Create Stack**.
6. Select the option to **Specify an Amazon S3 template URL** and paste the template link you copied.
7. Click **Next**.
8. On the **Specify Details** page, provide the required information using the *Configuration options* table as a guide.
9. Click **Next**.
10. **(Optional)** To disable rollback on failure for troubleshooting purposes, on the **Options** page, expand the **Advanced** heading and set **Rollback on failure** to **No**.
11. Click **Next**.
12. On the **Review** page, scroll to the bottom and select **I acknowledge that AWS CloudFormation might create IAM resources with custom names.**
13. Click **Create**.

You will be taken to the **CloudFormation - Stacks** page where you can monitor setup progress. When the setup is successful, the parent stack and all child stacks will show a status of **CREATE_COMPLETE**.

Deploy PingAccess into an existing VPC

You can deploy PingAccess into an existing VPC. Follow the same general steps as above, making the following modifications:

Note: To create a standalone base VPC, use the `create-base-vpc.yaml` template.

1. Use `pingaccess-setup.yaml` to specify the **Existing VPC ID** and create an **Export Name Prefix** for the PingAccess deployment. This step creates the link between the Export Name Prefix and the existing VPC.
2. Next, use `pingaccess-deploy.yaml` to configure and deploy PingAccess. Use the **Export Name Prefix** you specified in the previous step.

Configuration options

Stack name	The name of the parent stack. For example: `pingaccess-demo`.
Export name prefix	The prefix to be applied to all stack outputs imported and exported. For example: `pingaccess-stack`. You can create multiple clusters by using a unique **Export name prefix** for each.

Ping S3 repo stack name	The name of the stack used to establish the PingAccess software repository in S3. Use the **Stack Name** you specified in *Create the PingAccess software repository* on page 237. For example: `pingaccess-s3-stack`.
VPC name	The value applied to the VPC Name label. For example: `pingaccess-vpc`.
VPC CIDR block	The unique network CIDR block to be covered by the created VPC.
Number of public subnets specified	The number of public subnet CIDR blocks to be specified.
Public subnet CIDR blocks	A comma-delimited list of public subnet CIDR blocks. Must be unique subnets of VPC CIDR block.
Availability zones for public subnets	A comma-delimited list of Availability Zones per public subnet. At least two subnets must be in separate Availability Zones.
Number of private subnets specified	The number of private subnet CIDR blocks to be specified.
Private subnet CIDR blocks	A comma-delimited list of private subnet CIDR blocks. Must be unique subnets of VPC CIDR block.
Availability zones for private subnets	A comma-delimited list of Availability Zones per private subnet. At least two subnets must be in separate Availability Zones.
The allowed tenancy of instances launched into the VPC	The allowed tenancy of instances launched into the VPC.
Admin Node Private Host Name	The admin node host name to associate with the internal ELB.
Internal private domain name	The domain name to use for the private hosted zone for Ping Identity infrastructure internal DNS names. For example: `pingaccess.aws`.
Public hosted zone ID	The public hosted zone for use with external Route 53 DNS. You must have a public domain name registered with Amazon Route53. For more information, see *Route53 documentation*. For example: `my-aws-domain.com`.
Admin public host name	The fully qualified domain name to associate with the internet facing ELB for the admin API. This should be a unique subdomain of the specified Public hosted zone ID. This entry **must** be followed by a period. For example: `admin.my-aws-domain.com.`

Applications public host name	The fully qualified domain name to associate with the internet facing ELB for the engines. This should be a unique subdomain of the specified Public hosted zone ID. This entry **must** be followed by a period. For example: `app.my-aws-domain.com.`
PingAccess Admin API port	The TCP/IP port of the PingAccess Admin API listener. Use the default of `9000`.
PingAccess configuration query port	The TCP/IP port of the Admin Config Query listener. Use the default of `9090`.
SSH port	The TCP/IP port for the VPC bastion host and PingAccess instance SSH access. Use the default of `22`.
EC2 keypair name	The EC2 Key Pair for SSH access to the VPC bastion host and PingAccess instances. Use the key pair you created or imported.
Bastion host AMI ID	The Amazon Linux AMI ID for the bastion host. The default AMI is for US West (Oregon).
Bastion host instance type	The EC2 instance type for the bastion host. Use the default of `t2.micro` or higher.
Admin AMI ID	The Amazon Linux or RHEL 7.4 AMI ID for Admin instances. The default AMI is for US West (Oregon).
Admin instance type	The EC2 instance type for Admin nodes. Use the default of `m3.large` or higher.
Administrative node root volume name	The root volume device name for the specified Admin AMI. Commonly, Amazon Linux uses `/dev/xvda` and RHEL 7.4 uses `/dev/sda1`.
Administrative node root volume size	The size, in GB, for the root volume on the Admin instance. Use the default of 15 or higher.
Engine AMI ID	The Amazon Linux or RHEL 7.4 AMI ID for Engine node instances. The default AMI is for US West (Oregon).
Engine instance type	The EC2 instance type for Engine nodes. Use the default of `m3.large` or higher.
Engine node root volume name	The root volume device name for the specified Engine AMI. Commonly, Amazon Linux uses `/dev/xvda` and RHEL 7.4 uses `/dev/sda1`.
Engine node root volume size	The size, in GB, for the root volume on Engine instances. Use the default of 16 or higher.
Engine node root volume IOPS	The provisioned IOPS for the root volume on Engine instances. The maximum ratio of IOPS to volume size is 50:1.
Internal CA AMI ID	Amazon Linux AMI ID for the Internal CA. The default AMI is for US West (Oregon).

PingFederate AMI ID	Amazon Linux AMI ID for PingFederate. The default AMI is for US West (Oregon). This parameter is only used when ProvisionPingFederate is true.
Admin API certificate ID (ARN)	The ARN of the certificate to use with Admin API ELB listener. Use the certificate you created or imported.
Engines application certificate ID (ARN)	The ARN of the certificate to use with Engine API ELB listener. Use the certificate you created or imported.
Internal CA hostname	The unqualified hostname for the Internal CA within the private hosted zone.
Root CA Certificate TTL	The amount of time that the Root CA certificate will be valid before it expires. The default is 20 years (20y).
Certificate Generation Token TTL	Time to live for Vault tokens issued for certificate generation. The default is 1 year (1y).
Leaf Certificate Default TTL	The default time to live for generated leaf certificates before they expire. The default is 30 days (30d).
Leaf Certificate Max TTL	The maximum time to live allowed for leaf certificate generation requests.
Certificate Renewal Threshold	The number of days before a certificate expires that it should be renewed.
Certificate Renewal Schedule	The schedule expression that determines how often checks for certificate renewal should occur. For more information, see *AWS documentation for scheduled events*.
Provision PingFederate	Specify whether or not you want to provision a PingFederate instance. ▪ Select **True** to provision a PingFederate instance in the demonstration environment. ▪ Select **False** to use an existing PingFederate instance. If you select **False**, you must ensure your PingFederate configuration meets the requirements expected by the automation. To review the required PingFederate setup, see PingAccess for AWS: PingFederate environment requirements.
PingFederate Version	Indicates the PingFederate version expected in the S3 repository file name pingfederate-**`<PingFederateVersion>`** .zip. For example, pingfederate- **`8.4.4`** .zip. This field is ignored if **Provision PingFederate** is set to **False**.

PingFederate DNS Name	The fully qualified domain name to associate with the internet facing PingFederate instance. This should be a unique subdomain of the specified Public hosted zone ID. This entry **must** be followed by a period. For example: `pf.my-aws-domain.com.` This field is ignored if **Provision PingFederate** is set to **False**.
PingFederate Admin API Port	The PingFederate Admin API port. Use the default of `9999`. This field is ignored if **Provision PingFederate** is set to **False**.
PingFederate Runtime Port	The PingFederate Runtime port. Use the default of `9031`. This field is ignored if **Provision PingFederate** is set to **False**.
PingAccess Audit Logs Retention	The number of days to retain CloudWatch audit logs.
PingAccess Server Logs Retention	The number of days to retain CloudWatch server logs.
Lambda Logs Retention	The number of days to retain CloudWatch Lambda logs.
Java version	Indicates the Java JRE version expected in the S3 repository file name server-jre-8u **`<JavaVersion>`** -linux-x64.tar.gz. For example, server-jre-8u **`151`** -linux-x64.tar.gz.
PingAccess Version	Indicates the PingAccess version expected in the S3 repository file name pingaccess- **`<PingAccessVersion>`** .zip. For example, pingaccess- **`5.0.0`** .zip.

Configure Security Groups

To access the PingAccess and PingFederate Administration consoles, as well as the PingFederate runtime, AWS Security Groups must be configured to allow ingress traffic on the required ports. The complete installation provides a set of Security Groups that you must configure.

You can:

- *Configure the included security groups for ingress access*
- *Create a new Security Group*

Important: By default, all security groups created as part of this solution are configured to allow **all** egress traffic. Administrators may wish to review egress traffic requirements for their network and modify these settings. For more information, see *AWS documentation for Security Groups*.

Configure the included security groups for ingress access:

1. In AWS, navigate to **Services# Compute# EC2**.
2. In the left menu bar, under **Network & Security**, click **Security Groups**.

3. Select the Security Group you want to edit. See the table below for the list of groups.
4. In the bottom pane that appears, select the **Inbound** tab.
5. Click **Edit**.
6. In the **Edit inbound rules** window, click **Add Rule**.
    a. Add a **Custom TCP** rule.
    b. In the **Port Range** field, enter the port. See the table below for required ports.
    c. Enter the CIDR or IP for the machine you want to use for access. For quick access, you can select the **Source** of **My IP**.
7. Click **Save**.

PingAccess Admin API ingress	Port 9000
PingFederate	Ports 9999 and 9031
SSH Ingress (Type SSH)	Port 22

Note: Configuration of PingFederate Security Groups is only required if you are choosing to deploy with a demonstration PingFederate instance by selecting "True" for the ProvisionPingFederate parameter in the PingAccess for AWS CloudFormation template.

Create a new Security Group:

1. In AWS, navigate to **Services**# **Compute**# **EC2**.
2. Create a Security Group and assign it to the VPC that will include the PingAccess and PingFederate installations.
3. Add 3 **Custom TCP** rules and one **SSH** rule:
    - Port Range: **9000**, Source: **My IP**
    - Port Range: **9999**, Source: **My IP**
    - Port Range: **9031**, Source: **My IP**
    - Post Range: **22**, Source **My IP**
4. Click **Create**.

Access the PingAccess administration console

After the automated deployment is complete, you can access the PingAccess administrative console. The PingAccess administrative console will allow you to configure a token provider, create sites and applications, configure policies, and perform other related tasks.

Important: The PingAccess for AWS solution protects your admin node availability by ensuring a new instance of the admin node is created automatically in the event of a failure. This automated process deploys an instance of the PingAccess admin node based on the settings specified in `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`.

For this reason, it is **critical** to your deployment that you maintain this file. Any time changes are made to the PingAccess configuration, you **must** export the new configuration and replace the existing file at `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`. Failure to synchronize this file could lead to unexpected results if your admin node should fail for any reason.

Access the PingAccess Administration console

To access the PingAccess Administration console in your browser, use a combination of the Admin public host name and the PingAccess Administration port.

For example: `https://pa-admin.mypublichostname.com:9000`

For the demonstration environment, use the following credentials:

```
Username: joe
Password: 2Federate
```

If you are using an existing PingFederate instance, login with the credentials you created for use with the Simple Password Credential Validator in *PingAccess for AWS: PingFederate environment requirements*.

Access the PingFederate administration console

ⓘ **Note:** The following instructions only apply if you have chosen to deploy the PingAccess for AWS solution with a basic PingFederate instance for demonstration purposes.

After the automated deployment is complete, you can access the PingFederate administrative console. The PingFederate administrative console will allow you to configure clients and perform other related tasks.

Access the PingFederate Administration console

To access the PingFederate Administration Console in your browser, use a combination of the PingFederate public host name, the PingFederate Administration port, and the path to the PingFederate application.

For example: `https://pf-admin.mypublichostname.com:9999/pingfederate/app`

For this demonstration, use the default PingFederate credentials:

```
Username: Administrator
Password: 2Federate
```

Configure SSH connectivity

Configure SSH connectivity to the PingAccess Administration and Engine nodes so you can perform tasks such as viewing logs. The PingAccess Administration and Engine nodes are not exposed directly to the internet, but are configured to be placed behind a Bastion host through which you can access these resources. You first connect to the Bastion host, and then establish a connection to the desired resource.

The instructions provided here are intended to detail just one of the methods you can use to establish this connectivity. You may choose to use another method.

Prerequisites

This procedure requires that you have the following:

- The PingAccess private key
- The Public DNS hostname used by the Bastion host. Obtain this in EC2 by selecting the SSH Bastion instance and viewing the Description tab.
- The Private DNS hostname used by the PingAccess Admin Console and/or Engines. Obtain this in EC2 by selecting the desired instance and viewing the Description tab.

Configure SSH connectivity

1. Add the PingAccess private key to the SSH authentication agent. Run this command from the directory containing the private key. For example:

   ```
   ssh-add -K ~/.ssh/mypingaccesskeypair.pem
   ```

2. Add a host entry to your SSH configuration for the Bastion host. You will provide a name for the entry, the public DNS host name, and the user name used to access the host. For example, use the GNU nano editor to edit the host file with the following command:

   ```
   nano ~/.ssh/config
   ```

   Add an entry with the following information:

   ```
   Host aws-bastion
           Hostname mypublicbastiondns.compute.amazonaws.com
           User ec2-user
   ```

   Save the file.
3. Test the connection to the Bastion host:

   ```
   ssh aws-bastion
   ```
4. Add one of more SSH host entries for the Administration or Engine nodes, as required. Use the ProxyCommand function to route the connection through the Bastion host. For example:

   ```
   Host aws-console
           Hostname myinternaladmindns.compute.internal
           ProxyCommand ssh -q aws-bastion nc %h 22
           User ec2-user
   ```

   Save the file.

Establish a connection

Establish a connection to a configured PingAccess Administration or Engine node using the name you configured in the SSH host file. For example:

```
ssh aws-console
```

View PingAccess log files

To assist with troubleshooting, you can view PingAccess Admin and Engine log files. You can view logs related to the admin or engine nodes, and you can view *AWS CloudWatch* logs. To view node log files, you must first *establish an SSH connection* to the required node.

On an Admin or Engine node, the PingAccess installation can be found at `/usr/local/pingaccess`. Log files are contained in the `log` directory.

View log files

After you have established a connection to the node, navigate to the proper directory and open the file using an editor such as GNU nano. For example:

```
cd /usr/local/pingaccess/log
nano pingaccess.log
```

Maintaining the S3 bucket

To maintain the S3 bucket for updated files, simply delete the existing files and upload any new files using the steps found in *Populate the S3 bucket* on page 237.

Note: If you are planning to redeploy with an existing configuration, see *Redeploying with an existing configuration* on page 247.

Delete the PingAccess VPC

If removal of the PingAccess VPC is required, you can perform this task by deleting the top level stack. To do this:

1. In AWS, navigate to **Services# Management Tools# CloudFormation**.
2. In the CloudFormation console, select the top level template.
3. Click **Actions# Delete Stack**.
4. When prompted, confirm the operation.

ⓘ **Note:** If you are planning to redeploy with an existing configuration, see *Redeploying with an existing configuration* on page 247.

Redeploying with an existing configuration

You can redeploy the PingAccess for AWS solution using an existing configuration. You can specify to use an existing configuration by re-using the **Export name prefix**. Customers who wish to redeploy while maintaining an existing configuration should consider the following notes:

- You must first delete the PingAccess VPC
- When redeploying, you must use the same **Export name prefix**.
- Before you begin, you must delete the engine definitions present at `/automation-artifacts/<exportnameprefix>/pingaccess/engines`.
- If you intend to change certificate related parameters, you must delete the `/automation-artifacts/<exportnameprefix>/ca` folder.

Troubleshooting setup

For a variety of possible reasons, you can encounter errors or other issues during the creation of the PingAccess environment. This document is intended to provide guidance on troubleshooting some of these issues.

To troubleshoot setup, you should *disable Rollback on Failure* so that relevant logs and other helpful information remain available.

- *General*
- *Viewing logs on EC2 instances*
- *View CloudWatch logs*
- *View Lambda functions*
- *Modify templates to support Signature Version 4*

General

Some of the more common errors come from the planning and/or preparation process. For your chosen deployment type, ensure that you have provided and specified:

- Valid license files for PingAccess and, if applicable, PingFederate
- Valid installation packages for PingAccess and, if applicable, PingFederate
- Valid Server JRE install package
- A set of *AMIs* available to your AWS region that support Signature Version 2
- One or more *certificates* available to your AWS region
- If you are redeploying and reusing the same Export Name Prefix, ensure you have followed the advice in *Redeploying with an existing configuration* on page 247.

Viewing logs on EC2 instances

To view CloudFormation logs on your EC2 instances, *SSH to the instance* and view logs located at `/var/log`. CloudFormation logs begin with the `cfn-` prefix.

For instances that contain a PingAccess or PingFederate installation, you can view their respective log files at `/usr/local/pingaccess/log` and `/usr/local/pingfederate/log`.

View CloudWatch logs

CloudWatch logs sometimes contain helpful information in the event of a failure. View CloudWatch logs by navigating to **Services# Management Tools# CloudWatch** and clicking **Logs**.

You can filter logs for easier viewing. To find applicable CloudWatch logs, filter using the string `/pingidentity/{ExportNamePrefix}` where `{ExportNamePrefix}` is replaced by the **Export name prefix** you specified during *VPC creation*.

Click on a log group to view available log streams and their contents. To learn more about CloudWatch logs, see *AWS documentation for CloudWatch logs*.

View Lambda functions

To help narrow down causes for an issue, it may be helpful to view logged details for the lambda that is producing the error. To do this, navigate to **Services# Compute# Lambda** and filter by the VPC name. Locate and click the lambda function you are investigating. On the **Monitoring** tab, review the **Invocation errors** widget and click **Jump to Logs**.

Modify templates to support Signature Version 4

Some regions do not support Signature Version 2. In release 1.0, if you select a region that does not support Signature Version 2, you must to modify the templates to use Signature Version 4 style URLs. Since many templates are configured to use Signature Version 2 style URLs, the simplest way to make the required change is to find and replace through all files stored in the automation directory.

Find the string `https://${BUCKET_NAME}.s3.amazonaws.com/` and replace it with `https://s3-${AWS::Region}.amazonaws.com/${BUCKET_NAME}/`.

ⓘ **Note:** This workaround is not required in release 1.0.1 or above.

Configure an Application for PingAccess for AWS

Introduction

You can configure a protected application in the PingAccess for AWS environment. This document will provide instructions for configuring an application. These instructions are applicable for deployments that include a basic PingFederate instance, as well as deployments that make use of an existing PingFederate environment.

These instructions assume you have successfully deployed your environment and are able to access the PingAccess and PingFederate administration consoles.

ⓘ **Important:** The PingAccess for AWS solution protects your admin node availability by ensuring a new instance of the admin node is created automatically in the event of a failure. This automated process deploys an instance of the PingAccess admin node based on the settings specified in `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`.

For this reason, it is **critical** to your deployment that you maintain this file. Any time changes are made to the PingAccess configuration, you **must** export the new configuration and replace the existing file at `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`. Failure to synchronize this file could lead to unexpected results if your admin node should fail for any reason.

Protect an application

This document provides the steps required to protect an application using PingAccess and PingFederate in the AWS environment.

To protect an application, complete the following steps:

1. *Configure PingFederate* on page 249
2. *Configure a PingAccess web application* on page 250
3. *Configure a PingAccess API application* on page 252
4. *Test the web application* on page 253
5. *Test the API application* on page 253

Install and access the unprotected PingAccess QuickStart application

After the automated deployment is complete, you can install the PingAccess QuickStart application. The PingAccess QuickStart application can be used to test a protected PingAccess application.

Install the PingAccess QuickStart application

To install the PingAccess QuickStart application:

1. Download the PingAccess QuickStart application.
2. Unzip the file.
3. Copy `\pingaccess-quickstart-<version>\pf-dist\PingAccessQuickStart.war` to `<PingFedHome>\pingfederate\server\default\deploy`.
4. Restart PingFederate.

Access the unprotected PingAccess QuickStart application

To access the unprotected PingAccess QuickStart application in your browser, use a combination of the PingFederate public host name, the PingFederate runtime port, and the path to the QuickStart application.

For example: `https://pf-admin.mypublichostname.com:9031/PingAccessQuickStart`

Configure PingFederate

If you have deployed a basic PingFederate instance as part of the automation process, most of the configuration is complete. Likewise, if you have deployed with an existing PingFederate instance, and meet the PingFederate environment requirements, most of the required configuration is complete.

This document describes the steps required to create a client that will be used for the PingAccess Web Session you will assign to a web based application in PingAccess.

To create the web session client in PingFederate:

1. Log in to the PingFederate administration console.
2. Navigate to **Main# OAuth Settings# Clients** and click **Manage All**.
3. Click **Add Client**.
4. Specify a **Client ID**. For example:

   ```
   pa_wam
   ```
5. Specify a **Name**. For example:

   ```
   PingAccessWebAccessManagement
   ```
6. Select **Client Secret**.
7. Generate a secret by clicking **Generate Secret**. Copy this secret to a secure location so that you can use it in PingAccess configuration.
8. In the **Redirection URIs** field, add the OIDC callback redirect to the PingAccess server, for example:

   ```
   https://mypingaccessserver.my-aws-domain.com/pa/oidc/cb
   ```
9. Click **Add**.
10. Select the **Bypass Authorization Approval** check box.
11. For the **Allowed Grant Type** setting, select the **Authorization Code** check box.
12. Click **Save** (x2).

Configure a PingAccess web application

This document provides the steps required to configure the demo web application in PingAccess. In performing the steps required, you will:

1. *Create a web session* on page 250
2. *Create a virtual host* on page 250
3. *Create a site* on page 251
4. *Create an application* on page 251
5. *Configure resources*

Important: The PingAccess for AWS solution protects your admin node availability by ensuring a new instance of the admin node is created automatically in the event of a failure. This automated process deploys an instance of the PingAccess admin node based on the settings specified in `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`.

For this reason, it is **critical** to your deployment that you maintain this file. Any time changes are made to the PingAccess configuration, you **must** export the new configuration and replace the existing file at `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`. Failure to synchronize this file could lead to unexpected results if your admin node should fail for any reason.

Create a web session

Create a web session to control access. For more information, see *Web Sessions*.

1. Navigate to **Settings**# **Access**# **Web Sessions**
2. On the Web Session page, click **Add Web Session**.
3. Enter a unique **Name** for the web session, up to 64 characters, including special characters and spaces. For example:

```
PF_WAM
```

4. Select `Encrypted JWT` for **Cookie Type**.
5. Specify the **Audience** that the PA Token is applicable to, represented as a short, unique identifier between 1 and 32 characters. For example:

```
global
```

6. Specify the **OpenID Connect Login Type** `CODE`.
7. Specify the **Client ID**. For example, the Client ID you created is:

```
pa_wam
```

8. Specify the **Client Secret** you copied in *Configure PingFederate* on page 249.
9. Click **Save**.

Create a virtual host

Create a virtual host that PingAccess will respond to. For more information, see *Virtual Hosts*.

1. Navigate to **Settings**# **Access**# **Virtual Hosts**.
2. Click **Add Virtual Host**.
3. Enter the **Host** name for the Virtual Host or specify a wildcard. For example: `app.my-aws-pingaccess.com` or `*`.

4. Enter the **Port** number for the Virtual Host. In this environment, AWS is expecting a secure connection, so you must specify the secure port. For example:

```
443
```

5. Click **Save**.

Create a site

Create a site to define the location of the application that PingAccess is protecting. For more information, see *Sites*.

1. Navigate to **Main**# **Sites**# **Sites**.
2. Click **Add Site**.
3. Specify a **Name**. For example:

```
QuickStart
```

4. Specify the **Target**. The format for this is `hostname:port`. For example:

```
pf.my-pf-instance.com:9031
```

5. Select **Secure** and choose the `Trust Any` **Trusted Certificate Group**.
6. Click **Save**.

Note: If the target site cannot be contacted, the site is saved and a warning is displayed indicating the reason the site was not reachable.

Create an application

Create an application to define the resource that PingAccess is protecting. For more information, see *Applications*.

1. Navigate to **Main**# **Applications**.
2. Click **Add Application**.
3. Provide a unique name for the application. For example:

```
QuickStart
```

4. Specify the context at which the application is accessed at the site. For example:

```
/PingAccessQuickStart
```

5. Specify the virtual host for the application. For example:

```
*:443
```

6. Specify the application type `Web` and select the **Web Session** for the application. For example:

```
PF_WAM
```

7. Specify the application destination type Site and select the **Site** requests are sent to when access is granted. For example:

```
QuickStart
```

8. Select the **Require HTTPS** option.
9. Select the **Enabled** checkbox.
10. Click **Save**.

Configure resources

ⓘ **Note:** This step is only required for the demonstration environment. Your resource access requirements may differ.

Configure application resources to allow anonymous access to the root resource and authenticated access to the `/headers` resource.

1. Navigate to **Main#** **Applications**.
2. Expand the QuickStart application and click the edit button.
3. Select the **Resources** tab.
4. Expand the **Root Resource** and click the edit button.
5. Select the **Anonymous** checkbox and click **Save**.
6. Click **Add Resource**.
7. Enter a **Name** for the resource. For example: `headers`.
8. Enter the **Path Prefix** of `/headers`.
9. Click **Save**.

Configure a PingAccess API application

This document provides the steps required to configure the demo API application in PingAccess. Prior to completing these steps, ensure you have followed the steps in *Configure a PingAccess web application* on page 250 to create a **Virtual Host** and a **Site**.

ⓘ **Important:** The PingAccess for AWS solution protects your admin node availability by ensuring a new instance of the admin node is created automatically in the event of a failure. This automated process deploys an instance of the PingAccess admin node based on the settings specified in `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`.

For this reason, it is **critical** to your deployment that you maintain this file. Any time changes are made to the PingAccess configuration, you **must** export the new configuration and replace the existing file at `automation-artifacts/<export_prefix>/pingaccess/conf/pingaccess.json`. Failure to synchronize this file could lead to unexpected results if your admin node should fail for any reason.

Create an application

Create an application to define the resource that PingAccess is protecting. For more information, see *Applications*.

1. Navigate to **Main#** **Applications**.
2. Click **+ Add Application**.
3. In the **Name** field, enter a unique name for the application. For example:

   ```
   QuickStart
   ```
4. In the **Context Root** field, enter the context at which the application is accessed at the site. For example:

   ```
   /PingAccessQuickStart/api
   ```
5. In the **Virtual Host(s)** dropdown, specify the virtual host for the application. For example:

   ```
   *:443
   ```
6. In the **Application Type** dropdown, select `API`.
7. In the **Access Validation** dropdown, select **Token Provider**.

8. Specify the application destination type Site and select the **Site** requests are sent to when access is granted. For example:

   ```
   QuickStart
   ```
9. Select the **Require HTTPS** check box.
10. Select the **Enabled** check box.
11. Click **Save**.

Test the web application

This document describes the steps to test the demo web application.

Tip: For the most visible experience with this demo, perform these steps in a new private or incognito session to avoid the reuse of an existing session.

To test the web application:

1. Open your browser and navigate to the application using a combination of the **Applications public host name** you created in *Create the PingAccess VPC* on page 238 and the application Context Root. For example:

   ```
   https://app.my-aws-domain.com/PingAccessQuickStart
   ```
2. On the Web Access Management application, click **Try it Now**.
3. You will be prompted to authenticate. Enter the **Username** and **Password** combination included with the QuickStart demo application: `joe`/`2Federate`.

   The **headers** page is displayed.

Test the API application

This document describes the steps to test the demo API application.

Tip: For the most visible experience with this demo, perform these steps in a new private or incognito session to avoid the reuse of an existing session.

To test the demo API application:

1. Open your browser and navigate to the application using a combination of the **Applications public host name** you created in *Create the PingAccess VPC* on page 238 and the application Context Root. For example:

   ```
   https://app.my-aws-domain.com/PingAccessQuickStart
   ```
2. On the API Access Management application, click **Try it Now**.
3. Click the **Get Token** button.
4. You will be prompted to authenticate. Enter the **Username** and **Password** combination included with the QuickStart demo application: `joe`/`2Federate`.

   The token is obtained and pre-populated along with the URL context.
5. Click **Send API Request**.

   A **Response Code** of **200** indicates a successful test. **Response Headers** are displayed.

PingAccess for Azure AD

Solution overview

This document provides the steps required to configure PingAccess as part of the use case to provide secure external access to legacy on-premises applications using PingAccess for Azure AD and Microsoft Azure AD.

In this scenario, PingAccess provides an external path to legacy on-premises applications via the Azure AD Application Proxy through the use of header based authentication. Additionally, Microsoft Azure AD acts as the token provider for associated sessions.

PingAccess for Azure AD is a limited, free version of PingAccess for Microsoft Azure AD customers that provides protection for up to 20 applications.

This solution requires you to perform the following tasks. For more information about the requirements and options available for each task, review the task.

- *Configure PingAccess to use Azure AD as the token provider*
- *Configure a PingAccess application* for each application you want to protect and make available to Azure AD as part of this solution. Applications require configuration of:
  - A virtual host
  - A web session
  - An identity mapping
  - A site
  - An application

When the configuration is complete, you can test the application using the **Home Page URL** that you create in Azure AD.

Configure PingAccess to use Azure AD as the token provider

You can configure PingAccess to use Azure AD as the token provider.

Tip: For more information on configuring the token provider, see *Configure a token provider*.

Assumptions

- You have installed PingAccess and can access the *Administrative console*. For information on installing PingAccess, see *Install PingAccess*.

  Note: The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:

  ```
  Username: Administrator
  Password: 2Access
  ```

- If your administrative node uses a proxy for HTTP requests to the token provider, you have selected the HTTP Proxy in the **System# Clustering** section. See *Configure the primary administrative node* for more information.

Configure the token provider

1. Navigate to **Settings# System# Token Provider**.
2. In the **Issuer** field, enter the Microsoft Azure AD **Directory ID**. To obtain the Directory ID from Azure AD, in the Azure AD directory, navigate to **Manage# Properties** and copy the **Directory ID** value.
3. Provide a **Description** of the token provider.

4. In the **Trusted Certificate Group** list, select **Java Trust Store** or **Trust Any**.
5. Click **Save**.
6. To get the most out of the solution, configure *token provider specific options*.

Configure a PingAccess application

This document describes the steps required to configure PingAccess applications so that they are accessible to users via the Microsoft Azure ***MyApps*** portal. Use these instructions for each application that you want to configure.

Assumptions:

- You have installed PingAccess and can access the *Administrative console*. For information on installing PingAccess, see *Install PingAccess*.

  Note: The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:

  ```
  Username: Administrator
  Password: 2Access
  ```
- You have a *Microsoft Azure AD* Premium account for access to the Application Proxy feature.
- You have performed configuration in Microsoft Azure AD. For steps to configure Microsoft Azure AD, see *https://docs.microsoft.com/azure/active-directory/application-proxy-ping-access*.
- PingAccess is *configured* to use Azure AD as the token provider.

Create a virtual host

Tip: For more information on creating a virtual host, see *Create a virtual host*.

Important: In a typical configuration for this solution, you will create a virtual host for every application. For more information regarding configuration considerations and their impacts, see *Key considerations*.

1. Navigate to **Settings# Access# Virtual Hosts**.
2. Click **Add Virtual Host**.
3. In the **Host** field, enter the FQDN portion of the Azure AD **External URL**. For example, external URLs of `https://app-tenant.msappproxy.net/` and `https://app-tenant.msappproxy.net/AppName` will both demand a Host entry of `app-tenant.msappproxy.net`.
4. In the **Port** field, enter `443`.
5. Click **Save**.

Create a web session

Tip: For more information on creating a web session, see *Create a web session*.

1. Navigate to **Settings# Access# Web Sessions**.
2. Click **Add Web Session**.
3. Provide a **Name** for the web session.
4. Select the **Cookie Type**, either **Signed JWT** or **Encrypted JWT**.
5. Provide a unique value for the **Audience**.
6. In the **Client ID** field, enter the Azure AD **Application ID**.
7. In the **Client Secret** field, enter the **Key** you generated for the application in Azure AD.

8. **(Optional)** You can create and use custom claims with the Azure AD GraphAPI. If you choose to do so, click **Advanced** and deselect the **Request Profile** and **Refresh User Attributes** options. For more information on using custom claims, see *Optional - Use a custom claim*.
9. Click **Save**.

Create an identity mapping

ⓘ **Tip:** For more information on creating an identity mapping, see *Create an identity mapping*.

ⓘ **Note:** An identity mapping can be used with more than one application if more than one application is expecting the same data in the header.

1. Navigate to **Settings# Access# Identity Mappings**.
2. Click **Add Identity Mapping**.
3. Specify a **Name**.
4. Select the identity mapping **Type** of **Header Identity Mapping**.
5. In the **Attribute Mapping** table, specify the required mappings. For example:

Attribute Name	Header Name
upn	x-userprinciplename
email	x-email
oid	x-oid
scp	x-scope
amr	x-amr

6. Click **Save**.

ⓘ **Tip:** For more information about Azure AD tokens and for a list of supported claims, see *https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-token-and-claims*.

ⓘ **Important:** This use case only supports the claims discussed at the location included in the **Tip**.

Create a site

ⓘ **Tip:** For more information on creating a site, see *Create a site*.

ⓘ **Note:** In some configurations, it is possible that a site may contain more than one application. A site can be used with more than one application, where appropriate.

1. Navigate to **Main# Sites# Sites**.
2. Click **Add Site**.
3. Specify a **Name** for the site.
4. Enter the site **Target**. The target is the hostname:port pair for the server hosting the application. Do not enter the path for the application in this field. For example, an application at `https://mysite:9999/AppName` will have a target value of `mysite:9999`
5. Indicate whether or not the target is expecting **Secure** connections.
6. If the target is expecting secure connections, set the **Trusted Certificate Group** to **Trust Any**.
7. Click **Save**.

Create an application

You will create an application in PingAccess for each application in Azure that you want to protect.

Tip: For more information on creating an application, see *Create an application*.

1. Navigate to **Main# Applications**.
2. Click **Add Application**.
3. Specify a **Name** for the application.
4. Optionally, enter a **Description** for the application.
5. Specify the **Context Root** for the application. For example, an application at `https://mysite:9999/AppName` will have a context root of `/AppName`. If the application is on the root of the server, you can set the context root as `/`. The context root must begin with a slash (/), must not end with a slash (/), and can be more than one layer deep, for example, `/Apps/MyApp`.
6. Select the **Virtual Host** you created.

   Note: The combination of Virtual Host and Context Root must be unique in PingAccess.
7. Select the **Web Session** you created.
8. Select the **Site** you created that contains the application.
9. Select the **Identity Mapping** you created.
10. Select **Enabled** to enable the site when you save.
11. Click **Save**.

Getting started with PingAccess for Azure AD

This document provides information to get you started with the solution to protect legacy on-premises applications using **Microsoft Azure AD** and a limited version of PingAccess called **PingAccess for Azure AD**.

There are a variety of considerations that should be made when planning for a successful deployment.

Plan your deployment type and architecture

Use the *Deployment reference guide* to plan your deployment type and architecture. Learn about the differences between and benefits of a proxy deployment versus an agent based deployment, and decide to use one or a combination of both deployment types.

Design and plan a PingAccess cluster

Use the *Clustering reference guide* to design and plan your PingAccess cluster. For a high availability deployment, use a cluster that contains both a primary administrative node and a replica administrative node, along with additional engine nodes. For best performance, employ a *load balancing strategy*.

Install PingAccess

Ensure your systems meet the requirements so you can *Install PingAccess*.

Tune performance

Use the *Performance tuning reference guide* to configure your deployment for optimal performance.

Configure logging

Configure logging so that you can monitor your PingAccess deployment and troubleshoot application issues.

Configure the PingAccess token provider

Configure PingAccess to use Microsoft Azure AD as the *token provider*. Perform optional additional configuration that allows for communication with the *Azure AD Graph API*.

Configure applications

Configure applications to be made available by PingAccess to the Microsoft **MyApps portal** via Azure AD using the **Azure AD Application Proxy**.

Configure for dual internal and external secure access

Configure the solution so that applications are made securely available both externally through the Microsoft **MyApps** portal and internally through **PingAccess for Azure AD**.

PingAccess for Azure AD: Configure dual internal and external secure access

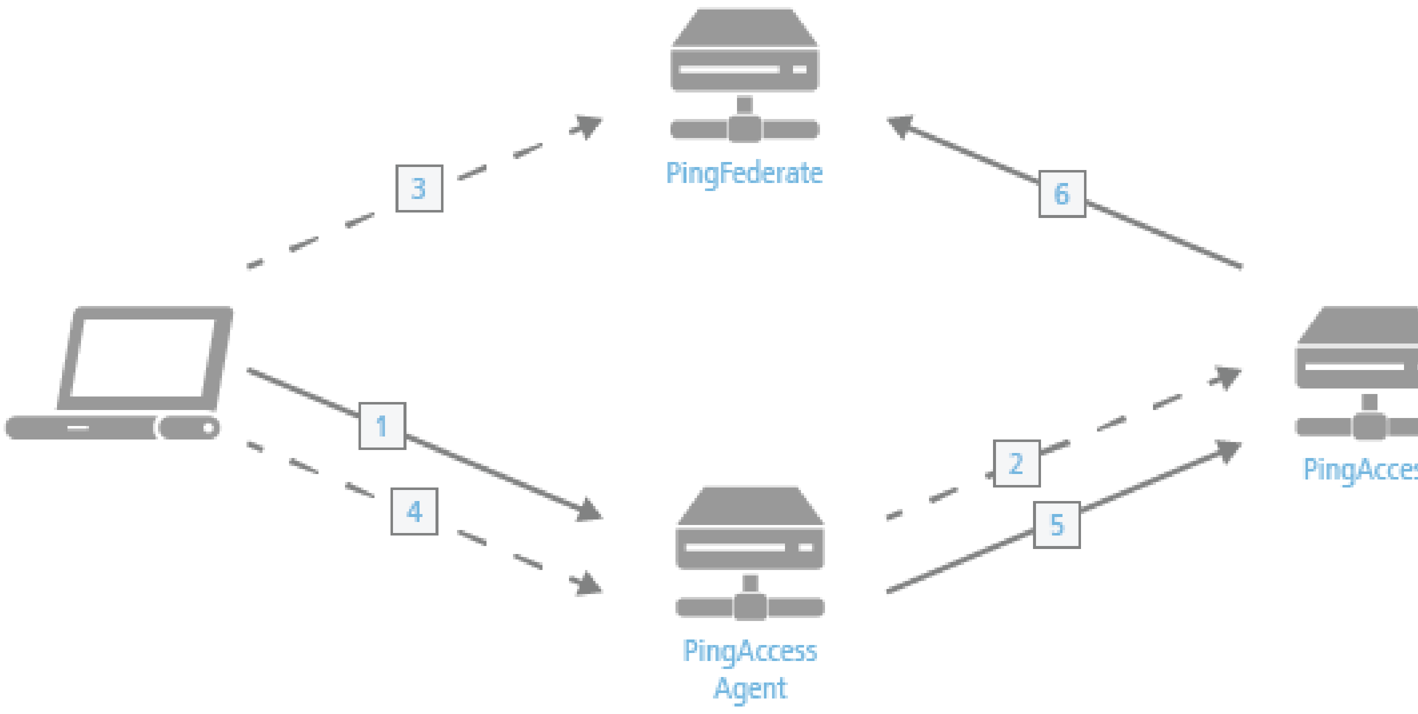
This document describes the steps necessary to configure applications for secure access both from inside the network and from outside.

1. Configure an application for secure external access using *Microsoft Azure AD* and *PingAccess for Azure AD*. Ensure that the application is functioning as expected.
2. In **PingAccess**, create a new *Virtual Host* that maps to the PingAccess host. For example, `<PingAccessServerName>:3000`.
3. Assign the new virtual host to the application in addition to the virtual host specified for Azure access.
4. In **Azure AD**, navigate to the App Registrations page and select the application.
5. Click **Reply URLs**, and add the internal PingAccess reply URL. For example, `<PingAccessServerName>:3000/pa/oidc/cb`.
6. Save the changes and test the configuration.

PingAccess Agent for Apache (RHEL)

Introduction

The PingAccess Agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.

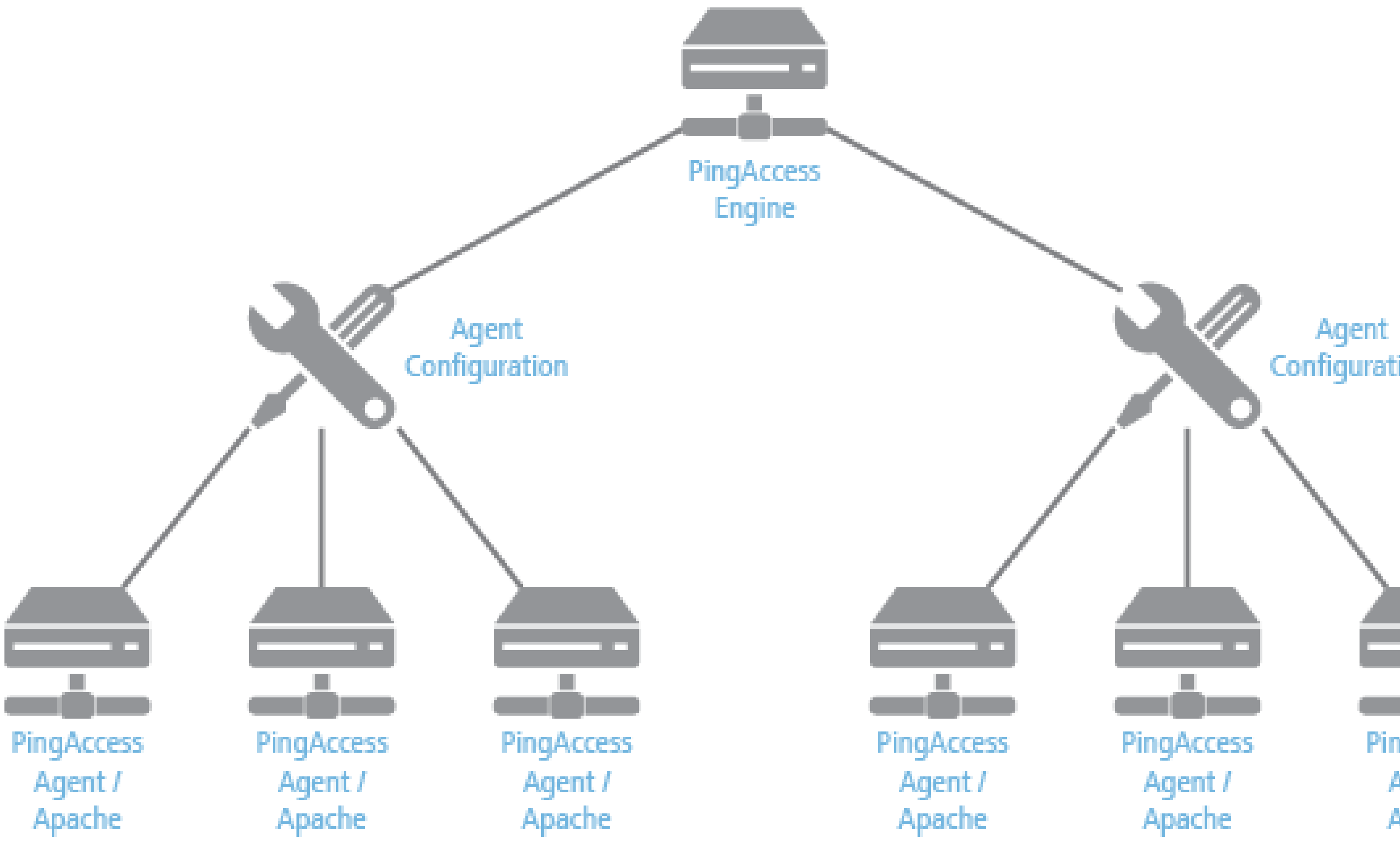


The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.

2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

Within the PingAccess Administration Console, agent nodes are configured with information that allows a PingAccess Agent to connect to the engine node to retrieve information about access control policies for resources within that agent's control. An agent configuration has a one-to-many relationship with PingAccess agents, allowing a single agent configuration bootstrap file to be used on multiple web servers within a server farm.


PingAccess Engine
Agent Configuration
Agent Configurati
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
Pin A A

ⓘ **Tip**: An agent node is a shared configuration used by one or more agents, rather than a specific agent instance.

System Requirements

The PingAccess Agent for Apache is supported on the following platforms:

- Apache HTTP Server 2.2 running on RedHat Enterprise Linux Server 6 (x86_64)
- Apache HTTP Server 2.2 running on RedHat Enterprise Linux Server 7 (x86 and x86_64)
- Apache HTTP Server 2.4 running on RedHat Enterprise Linux Server 6 (x86_64)
- Apache HTTP Server 2.4 running on RedHat Enterprise Linux Server 7 (x86_64)
- IBM HTTP Server 8.5 running on RedHat Enterprise Linux Server 6 (x86_64)
- IBM HTTP Server 9.0 running on RedHat Enterprise Linux Server 7 (x86_64)

As with any system that is reachable from the Internet, the server should be properly hardened. The PingAccess Agent for Apache includes an SELinux profile, and we recommend that SELinux be deployed on the server.

Installation

Installation - RHEL 6/Apache 2.2

Before you begin

ⓘ **Note:** For installation with the IBM HTTP Server on RedHat Enterprise Linux 6, see *Perform a manual installation with IBM HTTP Server* on page 267

Download and extract `pingaccess-agent-apache22-rhel6-<version>.zip`

ⓘ **Note:** The Agent RPM has required dependencies that may be available via standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `libpgm-5_2-0-5.2.122-32.1.x86_64.rpm`, `libsodium18-1.0.11-1.1.x86_64.rpm`, and `libzmq5-4.3.1-23.6.x86_64.rpm` packages.

About this task

To install the PingAccess Agent for Apache, perform the following steps:

Steps

1. In the PingAccess Console, go to the **Agents** page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
4. In RHEL, change to the `pingaccess-agent-apache22-rhel6/x86_64` directory.
5. As root, install the PingAccess Agent for Apache using the following command:

```
yum install pingaccess-agent-apache-*.rpm lib*.rpm
```

6. Copy the `<agentname>_agent.properties` file to `/etc/httpd/conf.d/agent.properties`.
7. As root, restart the Apache service using the following command:

```
service httpd restart
```

Manual Installation - RHEL 6/Apache 2.2

Before you begin

This procedure makes the following assumptions:

- The installation is being performed in a custom Apache instance run by a non-root user.
- The Apache installation is assumed to be installed at `$APACHE`. In the steps in this procedure, modify the paths specified based on where your Apache installation and configuration files are located.
- You have downloaded an `agent.properties` file. If you have not done so, you can do so using these steps:

1. In the PingAccess Console, navigate to the Sites & Agents page.

2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.

About this task
Use the following procedure to manually install the PingAccess Agent for Apache when Apache is installed in a non-standard way:

Steps

1. Install the following required dependencies from the RedHat Official Repositories:

   ```
   libcurl.x86_64
   pcre.x86_64
   ```

2. Copy the RPMs from the zip distribution into a directory called `pkgroot` and unpack them using the following commands:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   ```

3. Copy the extracted files to the appropriate places with the following commands:

   ```
   cp etc/httpd/conf.d/paa.conf $APACHE/conf
   cp -av usr/lib64/*.so* $APACHE/modules
   cp usr/lib64/httpd/modules/*.so $APACHE/modules
   ```

4. Add the following directive to the Apache configuration file (`$APACHE/conf/httpd.conf`) to include the PingAccess Agent for Apache module configuration:

   ```
   Include conf/paa.conf
   ```

5. Edit the `paa.conf` file and make the following changes:
   a. Add the following lines before the `LoadModule` directive:

      ```
      LoadFile modules/libpgm-5.2.so.0
      LoadFile modules/libsodium.so.18
      LoadFile modules/libzmq.so.5
      ```

   b. Change all occurrences of `conf.d` to `conf`.
6. Copy your downloaded `<hostname>_agent.properties` to `$APACHE/conf/agent.properties`.
7. Execute the command **`$APACHE/bin/apachectl restart`** to restart Apache.

Installation - RHEL 6/Apache 2.4

Before you begin

This procedure makes the following assumptions:

- The Apache configuration directory is: `$APACHE_ROOT/conf`
- The Apache modules directory is: `$APACHE_ROOT/modules`

Note: The Agent RPM has required dependencies that may be available via standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `libpgm-5_2-0-5.2.122-32.1.x86_64.rpm`, `libsodium18-1.0.11-1.1.x86_64.rpm`, and `libzmq5-4.3.1-23.6.x86_64.rpm` packages.

About this task

Steps

1. In the PingAccess Console, go to the **Agents** page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
4. In RHEL, download and extract `pingaccess-agent-apache24-rhel6-<version>.zip`.
5. Navigate to the `pingaccess-agent-apache24-rhel6` directory.
6. Edit the included `paa.conf` to modify the values for Apache's configuration and module directories.
7. Install the dependencies:

   ```
   yum install ./x86_64/openpgm*.rpm ./x86_64/zeromq3*.rpm
   ```
8. As root, copy the PingAccess Agent for Apache files to the appropriate places:

   ```
   cp ./x86_64/mod_paa.so $APACHE_ROOT/modules
   ```

   ```
   cp paa.conf $APACHE_ROOT/conf
   ```

   Note: By default, Apache on RHEL will automatically include all .conf files contained within `$APACHE_ROOT/conf` (using the IncludeOptional directive). If this has been disabled, add the following to Apache's httpd.conf:

   ```
   Include conf/paa.conf
   ```
9. Copy the `<agentname>_agent.properties` file to `$APACHE_ROOT/conf/agent.properties`
10. As root, restart the Apache service using the following command:

    ```
    $APACHE_ROOT/bin/apachectl restart
    ```

Manual Installation - RHEL 6/Apache 2.4

Before you begin

This procedure makes the following assumptions:

- You are installing the Agent in a custom Apache instance run by a non-root user.
- You can modify files and directories created by the Apache installer. If you installed Apache using yum, this procedure requires root access.
- The Apache installation is installed at `$APACHE`. In the steps in this procedure, modify the paths specified based on where your Apache installation and configuration files are located.
- You have downloaded an `agent.properties` file. If you have not done so, you can do so using these steps:

  1. In the PingAccess Console, navigate to the **Sites & Agents** page.
  2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
  3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.

About this task

Use the following procedure to manually install the PingAccess Agent for Apache when Apache is installed in a non-standard way:

Steps

1. Install the following required dependencies from the RedHat Official Repositories:

```
libcurl.x86_64
pcre.x86_64
```

2. Copy the Agent installation bundle to the target system and unpack it.
3. Change to the directory where you unpacked the Agent installation files.
4. Copy the extracted files to the appropriate places with the following commands:

```
cp paa.conf $APACHE/conf/
cd x86_64
cp mod_paa.so $APACHE/modules/
cp -av usr/lib64/*.so* $APACHE/modules/
```

5. Add the following directive to the Apache configuration file (`$APACHE/conf/httpd.conf`) to include the PingAccess Agent for Apache module configuration:

```
Include conf/paa.conf
```

6. Edit the `$APACHE/conf/paa.conf` file and make the following changes:
   a. Add the following lines before the `LoadModule` directive:

   ```
   LoadFile modules/libpgm-5.2.so.0
   LoadFile modules/libsodium.so.18
   LoadFile modules/libzmq.so.5
   ```

   b. Change all occurrences of `conf.d` to `conf`.
7. Copy your downloaded *`<agentname>`*`_agent.properties` to `$APACHE/conf/agent.properties`.
8. Execute the command **`$APACHE/bin/apachectl restart`** to restart Apache.

Installation - RHEL 7/Apache 2.2

Before you begin

This procedure makes the following assumptions:

- The Apache configuration directory is: `$APACHE_ROOT/conf`
- The Apache modules directory is: `$APACHE_ROOT/modules`

Note: The Agent RPM has required dependencies that may be available via standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `libpgm-5_2-0-5.2.122-32.1.x86_64.rpm`, `libsodium18-1.0.11-1.1.x86_64.rpm`, and `libzmq5-4.3.1-23.6.x86_64.rpm` packages.

Steps

1. In the PingAccess Console, go to the **Agents** page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.

3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
4. In RHEL, download and extract `pingaccess-agent-apache22-rhel7-<version>.zip`.
5. Navigate to the `pingaccess-agent-apache22-rhel7` directory.
6. Edit the included `paa.conf` to modify the values for Apache's configuration and module directories.
7. Install the dependencies:

   ```
   yum install ./x86_64/openpgm*.rpm ./x86_64/zeromq3*.rpm
   ```
8. As root, copy the PingAccess Agent for Apache files to the appropriate places:

   ```
   cp ./x86_64/mod_paa.so $APACHE_ROOT/modules
   ```

   ```
   cp paa.conf $APACHE_ROOT/conf
   ```

> **Note:** By default, Apache on RHEL will automatically include all .conf files contained within `$APACHE_ROOT/conf` (using the IncludeOptional directive). If this has been disabled, add the following to Apache's httpd.conf:
>
> ```
> Include conf/paa.conf
> ```

9. Copy the `<agentname>_agent.properties` file to `$APACHE_ROOT/conf/agent.properties`
10. As root, restart the Apache service using the following command:

    ```
    $APACHE_ROOT/bin/apachectl restart
    ```

Manual Installation - RHEL 7/Apache 2.2

Before you begin

This procedure makes the following assumptions:

- You are installing the Agent in a custom Apache instance run by a non-root user.
- You can modify files and directories created by the Apache installer. If you installed Apache using yum, this procedure requires root access.
- The Apache installation is assumed to be installed at `$APACHE`. In the steps in this procedure, modify the paths specified based on where your Apache installation and configuration files are located.
- You have downloaded an `agent.properties` file. If you have not done so, you can do so using these steps:

1. In the PingAccess Console, navigate to the **Sites & Agents** page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.

About this task
Use the following procedure to manually install the PingAccess Agent for Apache when Apache is installed in a non-standard way:

Steps

1. Install the following required dependencies from the RedHat Official Repositories:

   ```
   libcurl.x86_64
   pcre.x86_64
   ```
2. Copy the Agent installation bundle to the target system and unpack it.
3. Change to the directory where you unpacked the Agent installation files.

4. Copy the extracted files to the appropriate places with the following commands:

```
cp paa.conf $APACHE/conf/
cd x86_64
cp mod_paa.so $APACHE/modules/
cp -av usr/lib64/*.so* $APACHE/modules/
```

5. Add the following directive to the Apache configuration file (`$APACHE/conf/httpd.conf`) to include the PingAccess Agent for Apache module configuration:

```
Include conf/paa.conf
```

6. Edit the `$APACHE/conf/paa.conf` file and make the following changes:
   a. Add the following lines before the `LoadModule` directive:

   ```
   LoadFile modules/libpgm-5.2.so.0
   LoadFile modules/libsodium.so.18
   LoadFile modules/libzmq.so.5
   ```

   b. Change all occurrences of `conf.d` to `conf`.
7. Copy your downloaded *`<hostname>`*`_agent.properties` to `$APACHE/conf/agent.properties`.
8. Execute the command **`$APACHE/bin/apachectl restart`** to restart Apache.

Installation - RHEL 7/Apache 2.4

Before you begin

Note: For installation with the IBM HTTP Server on RedHat Enterprise Linux 7, see *Perform a manual installation with IBM HTTP Server* on page 267

Download and extract `pingaccess-agent-apache24-rhel7-`*`<version>`*`.zip`

Note: The Agent RPM has required dependencies that may be available via standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `libpgm-5_2-0-5.2.122-32.1.x86_64.rpm`, `libsodium18-1.0.11-1.1.x86_64.rpm`, and `libzmq5-4.3.1-23.6.x86_64.rpm` packages.

About this task

To install the PingAccess Agent for Apache, perform the following steps:

Steps

1. In the PingAccess Console, go to the **Agents** page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named *`<agentname>`*`_agent.properties`.
4. in RHEL, change to the `pingaccess-agent-apache24-rhel7/x86_64` directory.
5. As root, install the PingAccess Agent for Apache using the following command:

```
yum install pingaccess-agent-apache-*.rpm lib*.rpm
```

6. Copy the `<agentname>_agent.properties` file to `/etc/httpd/conf.d/agent.properties`.

7. As root, restart the Apache service using the following command:

   ```
   systemctl restart httpd.service
   ```

Manual Installation - RHEL 7/Apache 2.4

Before you begin

This procedure makes the following assumptions:

- The installation is being performed in a custom Apache instance run by a non-root user.
- The Apache installation is assumed to be installed at `$APACHE`. In the steps in this procedure, modify the paths specified based on where your Apache installation and configuration files are located.
- You have downloaded an `agent.properties` file. If you have not done so, you can do so using these steps:

1. In the PingAccess Console, navigate to the Sites & Agents page.
2. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
3. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.

About this task
Use the following procedure to manually install the PingAccess Agent for Apache when Apache is installed in a non-standard way:

Steps

1. Install the following required dependencies from the RedHat Official Repositories:

   ```
   libcurl.x86_64
   pcre.x86_64
   ```

2. Copy the RPMs from the zip distribution into a directory called `pkgroot` and unpack them using the following commands:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   ```

3. Copy the extracted files to the appropriate places with the following commands:

   ```
   cp etc/httpd/conf.modules.d/10-paa.conf $APACHE/conf
   cp -av usr/lib64/*.so* $APACHE/modules
   cp usr/lib64/httpd/modules/*.so $APACHE/modules
   ```

4. Add the following directive to the Apache configuration file (`$APACHE/conf/httpd.conf`) to include the PingAccess Agent for Apache module configuration:

   ```
   Include conf/10-paa.conf
   ```

5. Edit the `10-paa.conf` file and make the following changes:
   a. Add the following lines before the `LoadModule` directive:

      ```
      LoadFile modules/libpgm-5.2.so.0
      LoadFile modules/libsodium.so.18
      LoadFile modules/libzmq.so.5
      ```

   b. Change all occurrences of `conf.d` to `conf`.

6. Copy your downloaded `<hostname>_agent.properties` to `$APACHE/conf/agent.properties`.
7. Execute the command **`$APACHE/bin/apachectl restart`** to restart Apache.

Perform a manual installation with IBM HTTP Server

Before you begin

This procedure makes the following assumptions:

- The IBM HTTP Server has been installed and configured following IBM's documentation.
- The environment is running on RedHat Enterprise Linux 6 or RedHat Enterprise Linux 7.
- You have downloaded the appropriate `pingaccess-agent-apache22-*.zip` distribution file and have unzipped it.
- You have configured an agent in PingAccess, and have downloaded the `<agentname>_agent.properties` file.
- `apachectl` for the running IBM HTTP Server instance is in the path.
- The Apache installation is assumed to live at `$APACHE`. In the steps in this procedure, modify the paths specified based on where your Apache installation and configuration files are located.
- You have installed libcurl and PCRE or verified that they are installed. To install these packages, use this command:

  ```
  yum install libcurl pcre
  ```

About this task
Use the following steps to install the PingAccess Agent for Apache when using the IBM HTTP Server:

Steps

1. Change to the `pingaccess-agent-apache22-rhel<n>-<version>/<arch>/` directory.

   Note: Valid values for *<arch>* are `x86` for 32-bit and *x86_64* for 64-bit. 32-bit binaries are only available and supported on RHEL 7.

   For example: `cd pingaccess-agent-apache22-rhel7-1.2.0/x86/`
2. Extract the package RPMs with the command:

   ```
   mkdir pkgroot
   cp *.rpm pkgroot/
   cd pkgroot
   for r in *.rpm; do rpm2cpio $r | cpio -idmv; done
   ```
3. Execute the following command to copy the libraries to the appropriate Apache directories:
   - For RedHat Enterprise Linux 6 and 7 (x86_64):

     ```
     cp -av usr/lib64/*.so* $APACHE/modules
     ```
   - For RedHat Enterprise Linux 7 (x86):

     ```
     cp -av usr/lib/*.so* $APACHE/modules
     ```

4. Copy `mod_paa.so` into the Apache modules directory:
   - For RedHat Enterprise Linux 6:

     ```
     cp -av usr/lib64/httpd/modules/mod_paa.so $APACHE/modules
     ```
   - For RedHat Enterprise Linux 7:

     ```
     cp -av ../mod_paa.so $APACHE/modules
     ```
5. Copy `paa.conf` to the Apache configuration directory:
   - For RedHat Enterprise Linux 6:

     ```
     cp -av etc/httpd/conf.d/paa.conf $APACHE/conf
     ```
   - For RedHat Enterprise Linux 7:

     ```
     cp -av ../paa.conf $APACHE/conf
     ```
6. Edit `paa.conf` and add the following lines before the `LoadModule` directive:
   - For RedHat Enterprise Linux 6:

     ```
     LoadFile modules/libpgm-5.1.so.0
     LoadFile modules/libzmq.so.3
     ```
   - For RedHat Enterprise Linux 7:

     ```
     LoadFile modules/libpgm-5.2.so.0
     LoadFile modules/libzmq.so.3
     ```
7. In `paa.conf`, update the values for `PaaPropertyFiles` and `PaaCertificateDir` to point to your Apache `conf` directory.
8. Add the following directive to the Apache configuration file (`$APACHE/conf/httpd.conf`) to include the PingAccess Agent for Apache module configuration:

   ```
   Include conf/paa.conf
   ```
9. Copy the `<agentname>_agent.properties` file to `$APACHE/conf/agent.properties`.
10. Restart the Apache service by running `apachectl restart`.

Uninstall
You can remove the PingAccess Agent from an RHEL system.

About this task

If you installed the Ping Access Agent using the standard installation process, you can uninstall it with this command:

```
sudo yum remove pingaccess-agent-rhel.x86_64
```

If you installed the Ping Access Agent manually with an IBM HTTP Server, you can uninstall it with this procedure:

Steps

1. Remove the `$APACHE/conf/agent.properties` file:

   ```
   rm $APACHE/conf/agent.properties
   ```

2. Remove the following directive from the Apache configuration file (`$APACHE/conf/httpd.conf`):

   ```
   Include conf/paa.conf
   ```

3. Remove `paa.conf` from the `$APACHE/modules` directory:

   ```
   rm $APACHE/modules/paa.conf
   ```

4. Remove all `.so` files from the `$APACHE/modules` directory:

   ```
   rm $APACHE/modules/*.so*
   ```

5. Restart the Apache service by running **`apachectl restart`**.

Configuration

The agent configuration is managed through the `paa.conf` and `agent.properties` configuration files.

The `/etc/httpd/conf.d/paa.conf` file contains these configuration options:

Parameter	Definition	Default Value
PaaCertificateDir	String value containing the path to the certificates extracted from the `.properties` files.	`conf.d`

Parameter	Definition	Default Value

PaaEnabled	Determines whether the agent is enabled or disabled for a specific server configuration. Valid values: `on/off`
	
	This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The most specific value is used.
	
	Note: If you disable the PaaEnabled parameter globally, ensure that the PaaEnabled directive is set to `on` for the PingAccess reserved application context root. This is `/pa` by default.
	
	For example, adding this text to an included configuration file enables PingAccess for the `/pa` context root and for the `/var/www/html/one` directory:
	
	`<VirtualHost *:81>`
	`    <Location /pa>`
	`        PaaEnabled on`
	`    </Location>`
	`    <Directory "/var/www/`
	`html/one">`
	`        PaaEnabled on`
	`    </Directory>`
	`</VirtualHost>`
	
	Adding this text to an included configuration file disables PingAccess for all content in the `/var/www/html/two` directory except for files named `page2.html`.
	
	`<VirtualHost *:81>`
	`    <Directory "/var/www/`
	`html/two">`
	`        PaaEnabled off`
	`       <Files "page2.html">`
	`            PaaEnabled on`
	`       </Files>`
	`    </Directory>`
	`</VirtualHost>`
		`on`

PaaPropertyFiles	List of `.properties` files which store configuration data used to connect the agent to the PingAccess engine nodes the agent will communicate with.	`conf.d/agent.properties`

Parameter	Definition	Default Value
PaaEnabledNoteName	An optional parameter which defines a note name. If a request includes a note with this name and a value of `on` or `off`, this value overrides the PaaEnabled setting for that request. If you want to use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

The configured `agent.properties` files can contain the following parameters:

Parameter	Definition	Default Value
agent.engine.configuration.scheme	The URI scheme used to connect to the engine node. Valid values are `http` and `https`.	https
agent.engine.configuration.host	The PingAccess hostname.	The value in the Agent Node's `PingAccess Host` field.
agent.engine.configuration.port	The port the agent connects to on the PingAccess host. This value is defined in the PingAccess `run.properties` file.	Defined in the PingAccess Admin UI
agent.engine.configuration.username	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess Admin UI
agent.engine.configuration.shared.secret	The password used to authenticate the agent to the engine.	Defined in the PingAccess Admin UI
agent.engine.configuration.bootstrap.truststore	The base64-encoded public certificate used to establish HTTPS trust by the agent to the PingAccess engine. **Note:** If you are having difficulty connecting an agent to the PingAccess engine, verify that the Agent Trusted Certificate has been configured correctly in *Agent Management*.	Generated by PingAccess
agent.engine.configuration.maxConnections	The number of connections a single web server worker process maintains to the PingAccess engine defined in the agent.engine.configuration.host parameter.	10
agent.engine.configuration.timeout	The maximum time (in milliseconds) a request to PingAccess can take from the agent. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000

Parameter	Definition	Default Value
agent.engine.configuration.connectTimeout	The maximum time (in milliseconds) the agent can take to connect to the PingAccess engine. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.cache.missInitialTimeout	The maximum time (in milliseconds) a web server worker process waits for a response to a policy cache request sent to other web server worker processes.	5
agent.cache.broker.publisherPort	The network port web server processes use to publish policy cache requests to other web server worker processes. This port is bound to the localhost network only.	3031
agent.cache.broker.subscriberPort	The network port web server processes use to receive policy cache requests from other web server worker processes. This port is bound to the localhost network only.	3032
agent.cache.maxTokens	The maximum number of tokens stored in the policy cache for a single web server worker process. A value of `0` means there is no maximum.	0
agent.cache.disabled	Determines whether caching of policy decisions is enabled or disabled. A value of `1` disables caching, forcing the agent to communicate with the PingAccess host any time a policy decision needs to be made. **Warning:** Disabling caching has a significant impact on the scalability of the PingAccess Policy servers, as every rule evaluation is processed by the Policy Server. This option should only be used as a last resort because of the performance penalty.	0

Parameter	Definition	Default Value
agent.cache.type	Controls the type of policy cache used by the agent. There are three valid values for this property: ▪ **AUTO** - The AUTO cache type determines the appropriate cache to use based on the number of worker processes. If the number of worker processes is 1, the agent uses the STANDALONE cache. If the number of worker processes is 2 or more, the agent uses the ZMQ cache. ▪ **STANDALONE** - The STANDALONE cache type does not share policy cache entries across worker processes. ▪ **ZMQ** - The ZMQ cache type allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	AUTO

Changes to the `agent.properties` file require a restart of the web server.

Tip: See the *Performance Tuning Guide* for a discussion on improving agent performance.

Adding trust for an anchor certificate
You can add trust for an anchor certificate.

Steps

1. Use the **`certutil`** command to add certificates to the `cert9.db` file. For example:

```
certutil -L -d sql: /etc/pki/nssdb
```

2. Clear the `agent.engine.configuration.bootstrap.truststore` property in `agent.properties` to have the agent fallback to the system truststore (`/etc/pki/nssdb` in the example).

Log Configuration

The PingAccess Agent for Apache writes its information to the standard Apache error log, defined in the Apache configuration with the `ErrorLog` configuration directive. All information logged by the PingAccess Agent is prefaced with the string `[paa]`. PingAccess Agent monitoring and performance information is prefaced with the string `[paa-monitoring]`, and contains information about how long the PingAccess Agent took to fill a cache request and how long the total policy decision took.

The LogLevel used by the PingAccess Agent module is taken from the top-level httpd.conf configuration.

Troubleshooting

The following table lists some potential problems and resolutions you might encounter with the PingAccess Agent for RHEL.

Issue	Resolution
Agent receives an unknown protocol error when attempting to contact the administrative node	This can indicate that the operating system is using sha1 for encryption. This protocol is no longer supported by default in PingAccess. We recommend switching to sha256. If you cannot switch to sha256, you can re-enable sha1: **1.** Open the `run.properties` file. **2.** Add TLSv1 to the protocol list. For example: `tls.default.protocols=`**TLSv1,** `TLSv1.1, TLSv1.2` **3.** Add the SHA entries to the cipher suites list. For example: `tls.default.cipherSuites = TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\ TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,\ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,\ TLS_RSA_WITH_AES_128_GCM_SHA256,\ TLS_RSA_WITH_AES_128_CBC_SHA256,\ TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,\ TLS_EMPTY_RENEGOTIATION_INFO_SCSV, \` **TLS_RSA_WITH_AES_128_CBC_SHA, \ TLS_DHE_RSA_WITH_AES_128_CBC_SHA, \ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, \ TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, \ TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA**

Release Notes

Version History

- **Version 1.4** – June 2019

  Agent SDK for C version 1.2.0

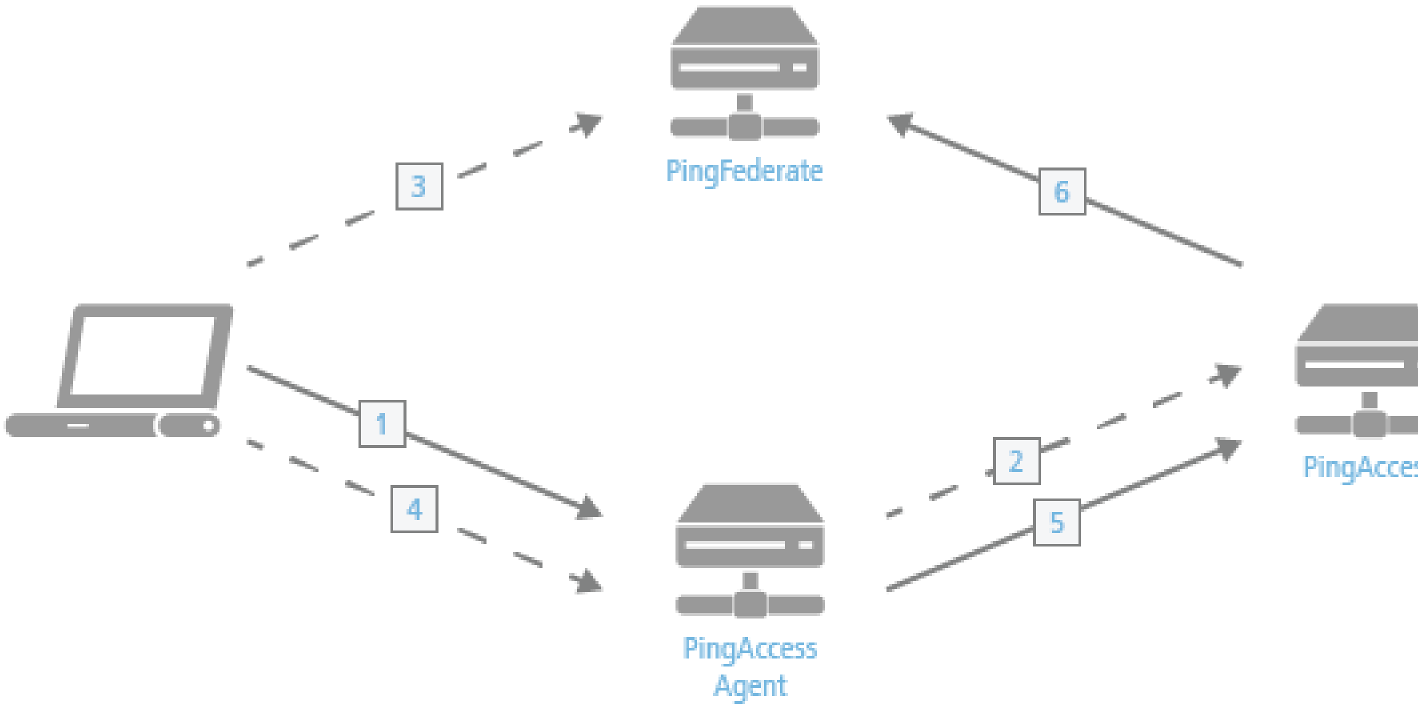
  - The PAAEnabled directive can now be used inside a directory or location container
  - Added ability to set policy caching mechanism using a property in the `agent.properties` file
  - Added ability to enable or disable agent processing for a request based on a note field
  - Fixed a potential security issue
- **Version 1.3.2** – November 2018
  - Fixed a potential security issue
- **Version 1.3** – February 2017
  - Added support for Apache 2.4 on RHEL 6
  - The Agent can now be disabled for specific hosts using the new configuration option: `PaaEnabled`
- **Version 1.2** – May 2016
  - Added support for IBM HTTP Server
- **Version 1.1** – December 2014
  - Added Support for Apache 2.4 on RedHat Enterprise Linux 7
  - Corrected a potential security issue related to caching (SECBL007). This security bulletin is available in the Ping Identity Support Portal (*http://ping.force.com/Support*).

- **Version 1.0** – July 2014
  - Initial Release

PingAccess Agent for Apache (SLES)

Introduction

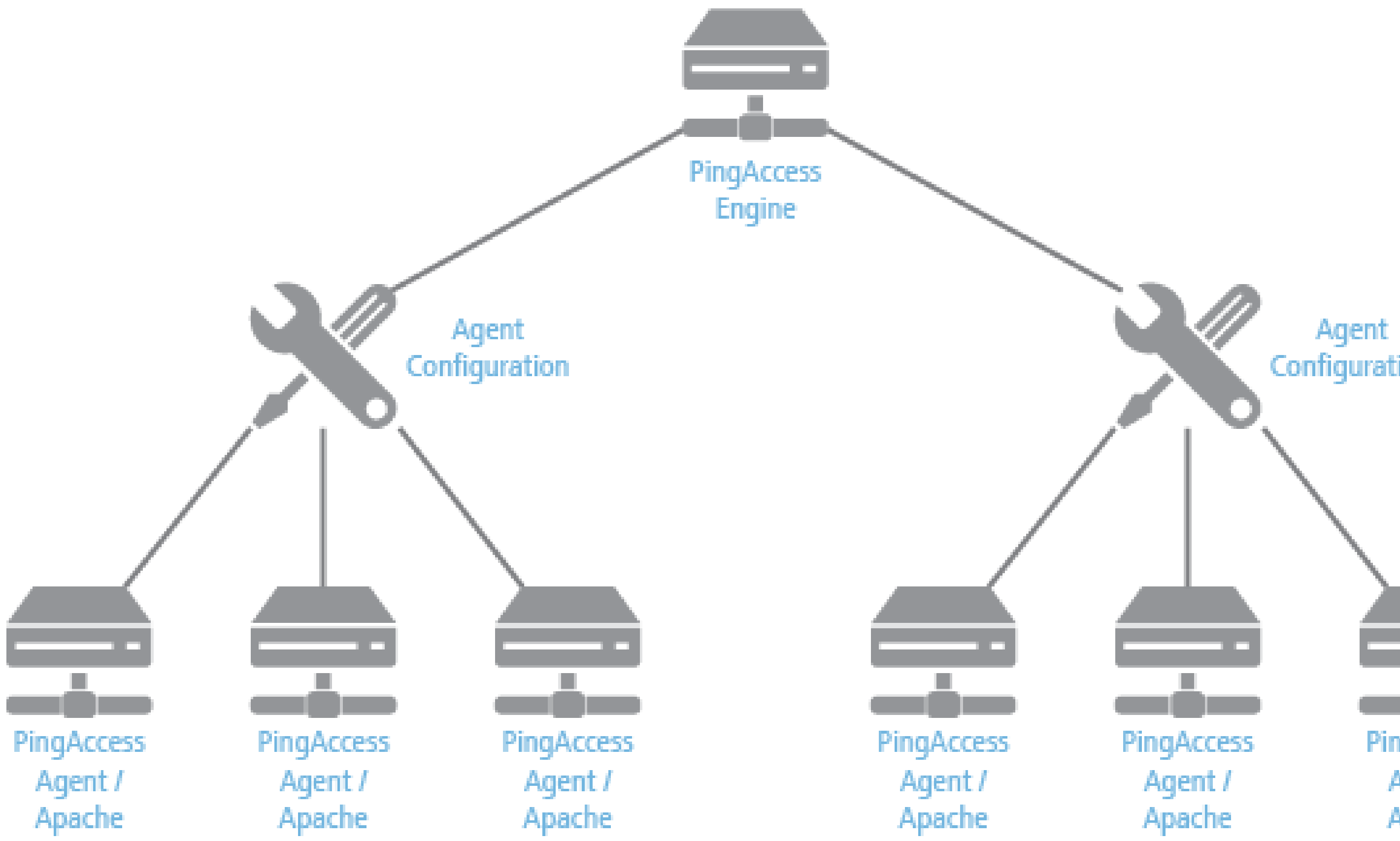
The PingAccess Agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

Within the PingAccess Administration Console, agent nodes are configured with information that allows a PingAccess Agent to connect to the engine node to retrieve information about access control policies for resources within that agent's control. An agent configuration has a one-to-many relationship with PingAccess agents, allowing a single agent configuration bootstrap file to be used on multiple web servers within a server farm.


PingAccess Engine
Agent Configuration
Agent Configurati
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
Pin
A
A

> ⓘ **Tip:** An agent node is a shared configuration used by one or more agents, rather than a specific agent instance.

System Requirements

The PingAccess Agent for Apache is supported on the following platforms:

- Apache HTTP Server 2.2 running on SUSE Linux Enterprise Server 11 SP4 (x86_64)
- Apache HTTP Server 2.4 running on SUSE Linux Enterprise Server 12 SP2 (x86_64)

As with any system that is reachable from the Internet, the server should be properly hardened. The PingAccess Agent for Apache includes an SELinux profile, and we recommend that SELinux be deployed on the server.

Installation

Before you begin

This procedure makes the following assumptions:

- The Apache configuration directory is: `/etc/apache2/conf.d`
- The Apache modules directory is: `/usr/lib64/apache2`

For custom installations:

- Modify the configuration and module paths below as needed.
- Edit the included `paa.conf` to modify the values for Apache's configuration and module directories.

To install the PingAccess Agent for Apache, perform the following steps:

Steps

1. Download and extract `pingaccess-agent-apache<version>.zip`.
2. Navigate to the `pingaccess-agent-apache<version>` directory.
3. If you are installing on SLES 12, import the gpg key:

   ```
   rpm --import https://download.opensuse.org/repositories/network:/
   messaging:/zeromq:/release-stable/SLE_12_SP4/repodata/repomd.xml.key
   ```
4. Install the dependencies:

   ```
   zypper in ./x86_64/lib*.rpm
   ```
5. As root, copy the PingAccess Agent for Apache files to the appropriate places:

   `cp ./x86_64/mod_paa.so /usr/lib64/apache2`

   `cp paa.conf /etc/apache2/conf.d`

   Note: By default, Apache on SLES will automatically include all `.conf` files contained within `conf.d` (using the IncludeOptional directive). If this has been disabled, add the following to Apache's `httpd.conf`:

   `Include /etc/apache2/conf.d/paa.conf`
6. In the PingAccess Console, go to the **Agents** page.
7. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
8. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
9. Copy the `<agentname>_agent.properties` file to `/etc/apache2/conf.d/agent.properties`
10. As root, restart the Apache service using one of the following commands:

    `rcapache2 restart`

    `$APACHE_ROOT/bin/apachectl restart`

Uninstall
You can remove the PingAccess Agent from a SUSE Linux system.

Steps

Run this command:

```
sudo zypper rm pingaccess-agent-suse.x86_64
```

Configuration

The agent configuration is managed through the `paa.conf` and `agent.properties` configuration files.

The `/etc/httpd/conf.d/paa.conf` file contains these configuration options:

Parameter	Definition	Default Value
PaaCertificateDir	String value containing the path to the certificates extracted from the `.properties` files.	`conf.d`

Parameter	Definition	Default Value

PaaEnabled	Determines whether the agent is enabled or disabled for a specific server configuration. Valid values: `on/off`
	
	This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The most specific value is used.
	
	Note: If you disable the PaaEnabled parameter globally, ensure that the PaaEnabled directive is set to `on` for the PingAccess reserved application context root. This is `/pa` by default.
	
	For example, adding this text to an included configuration file enables PingAccess for the `/pa` context root and for the `/var/www/html/one` directory:
	
	`<VirtualHost *:81>`
	`    <Location /pa>`
	`        PaaEnabled on`
	`    </Location>`
	`    <Directory "/var/www/html/one">`
	`        PaaEnabled on`
	`    </Directory>`
	`</VirtualHost>`
	
	Adding this text to an included configuration file disables PingAccess for all content in the `/var/www/html/two` directory except for files named `page2.html`.
	
	`<VirtualHost *:81>`
	`    <Directory "/var/www/html/two">`
	`        PaaEnabled off`
	`       <Files "page2.html">`
	`            PaaEnabled on`
	`       </Files>`
	`    </Directory>`
	`</VirtualHost>`
		`on`

Parameter	**Definition**	**Default Value**
PaaPropertyFiles	List of .properties files which store configuration data used to connect the agent to the PingAccess engine nodes the agent will communicate with.	`conf.d/agent.properties`
PaaEnabledNoteName	An optional parameter which defines a note name. If a request includes a note with this name and a value of `on` or `off`, this value overrides the PaaEnabled setting for that request. If you want to use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

ⓘ **Note:** It is not necessary to make any changes to paa.conf if the steps in the *Installation* section were followed.

The configured `agent.properties` files can contain the following parameters:

Parameter	**Definition**	**Default Value**
agent.engine.configuration.scheme	The URI scheme used to connect to the engine node. Valid values are `http` and `https`.	https
agent.engine.configuration.host	The PingAccess hostname.	The value in the Agent Node's `PingAccess Host` field.
agent.engine.configuration.port	The port the agent connects to on the PingAccess host. This value is defined in the PingAccess `run.properties` file.	Defined in the PingAccess Admin UI
agent.engine.configuration.username	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess Admin UI
agent.engine.configuration.shared.secret	The password used to authenticate the agent to the engine.	Defined in the PingAccess Admin UI
agent.engine.configuration.bootstrap.truststore	The base64-encoded public certificate used to establish HTTPS trust by the agent to the PingAccess engine. ⓘ **Note:** If you are having difficulty connecting an agent to the PingAccess engine, verify that the Agent Trusted Certificate has been configured correctly in *Agent Management*.	Generated by PingAccess

Parameter	**Definition**	**Default Value**
agent.engine.configuration.maxConnections	The number of connections a single web server worker process maintains to the PingAccess engine defined in the agent.engine.configuration.host parameter.	10
agent.engine.configuration.timeout	The maximum time (in milliseconds) a request to PingAccess can take from the agent. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.engine.configuration.connectTimeout	The maximum time (in milliseconds) the agent can take to connect to the PingAccess engine. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.cache.missInitialTimeout	The maximum time (in milliseconds) a web server worker process waits for a response to a policy cache request sent to other web server worker processes.	5
agent.cache.broker.publisherPort	The network port web server processes use to publish policy cache requests to other web server worker processes. This port is bound to the localhost network only.	3031
agent.cache.broker.subscriberPort	The network port web server processes use to receive policy cache requests from other web server worker processes. This port is bound to the localhost network only.	3032
agent.cache.maxTokens	The maximum number of tokens stored in the policy cache for a single web server worker process. A value of 0 means there is no maximum.	0

Parameter	Definition	Default Value
agent.cache.disabled	Determines whether caching of policy decisions is enabled or disabled. A value of `1` disables caching, forcing the agent to communicate with the PingAccess host any time a policy decision needs to be made. **Warning:** Disabling caching has a significant impact on the scalability of the PingAccess Policy servers, as every rule evaluation is processed by the Policy Server. This option should only be used as a last resort because of the performance penalty.	0
agent.cache.type	Controls the type of policy cache used by the agent. There are three valid values for this property: ▪ **AUTO** - The AUTO cache type determines the appropriate cache to use based on the number of worker processes. If the number of worker processes is 1, the agent uses the STANDALONE cache. If the number of worker processes is 2 or more, the agent uses the ZMQ cache. ▪ **STANDALONE** - The STANDALONE cache type does not share policy cache entries across worker processes. ▪ **ZMQ** - The ZMQ cache type allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	AUTO

Changes to the `agent.properties` file require a restart of the web server.

Tip: See the *Performance Tuning Guide* for a discussion on improving agent performance.

Adding trust for an anchor certificate

You can add trust for an anchor certificate.

Steps

1. Use the **`certutil`** command to add certificates to the `cert9.db` file. For example:

   ```
   certutil -L -d sql: /etc/pki/nssdb
   ```

2. Clear the `agent.engine.configuration.bootstrap.truststore` property in `agent.properties` to have the agent fallback to the system truststore (`/etc/pki/nssdb` in the example).

Log Configuration

The PingAccess Agent for Apache writes its information to the standard Apache error log, defined in the Apache configuration with the `ErrorLog` configuration directive. All information logged by the PingAccess Agent is prefaced with the string `[paa]`. PingAccess Agent monitoring and performance information is prefaced with the string `[paa-monitoring]`, and contains information about how long the PingAccess Agent took to fill a cache request and how long the total policy decision took.

The LogLevel used by the PingAccess Agent module is taken from the top-level httpd.conf configuration.

Troubleshooting

The table below lists some potential problems and resolutions you may encounter with the PingAccess Agent for SLES.

Issue	Resolution

Agent receives an unknown protocol error when attempting to contact the administrative node	This can indicate that the operating system is using sha1 for encryption. This protocol is no longer supported by default in PingAccess.
	
	We recommend switching to sha256. If you cannot switch to sha256, you can re-enable sha1:
	
	1. Open the `run.properties` file.
	2. Add TLSv1 to the protocol list. For example:
	
	`tls.default.protocols=`**TLSv1,** `TLSv1.1, TLSv1.2`
	
	3. Add the SHA entries to the cipher suites list. For example:
	
	`tls.default.cipherSuites =`
	`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_RSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_EMPTY_RENEGOTIATION_INFO_SCSV, \`
	TLS_RSA_WITH_AES_128_CBC_SHA, \
	TLS_DHE_RSA_WITH_AES_128_CBC_SHA, \
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, \
	TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, \
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

Release Notes

Version History

- **Version 1.4** – June 2019

  Agent SDK for C version 1.2.0

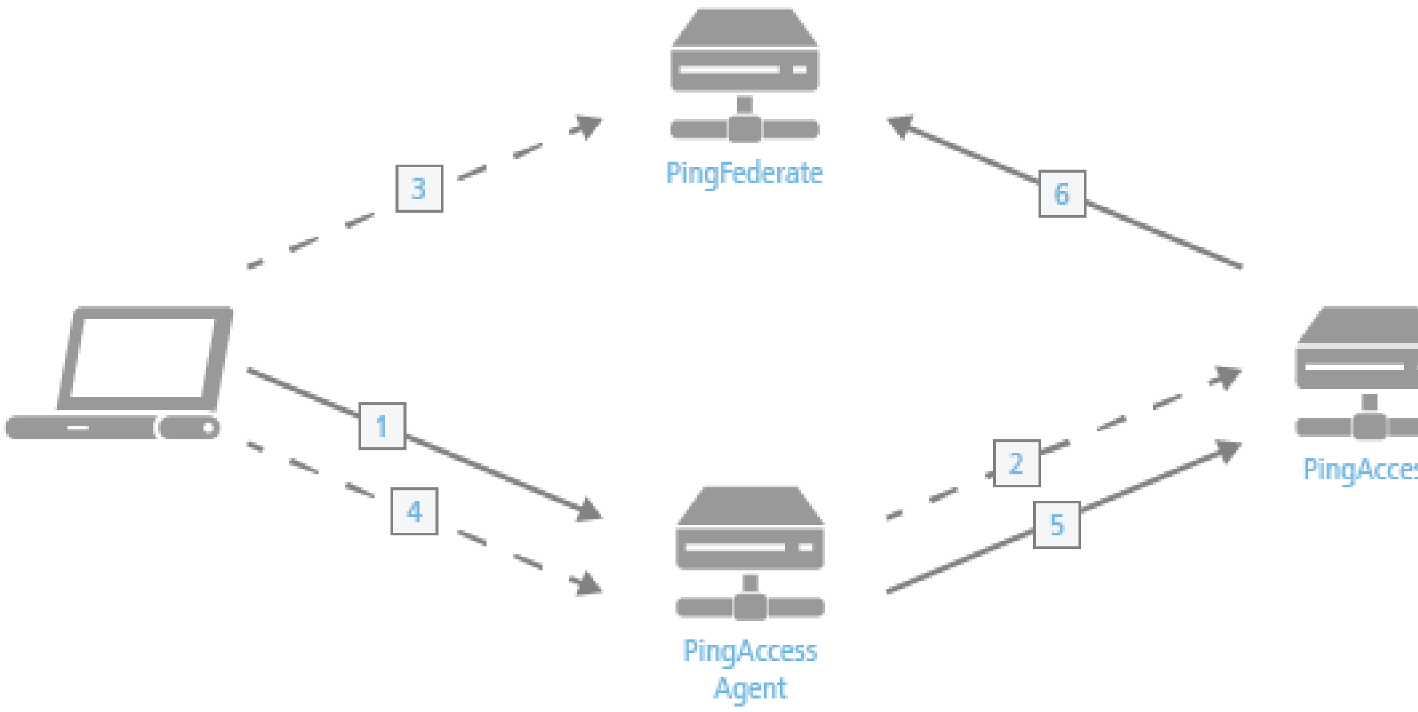
  - The PAAEnabled directive can now be used inside a directory or location container
  - Added ability to set policy caching mechanism using a property in the `agent.properties` file
  - Added ability to enable or disable agent processing for a request based on a note field
  - Fixed a potential security issue
- **Version 1.3.2** – November 2018
  - Fixed a potential security issue
- **Version 1.3** – March 2017
  - Initial release for Apache 2.2 on SLES 11 and Apache 2.4 on SLES 12

Note: Version is aligned with PingAccess Agent for Apache (RHEL)

PingAccess Agent for Apache (Windows)

Introduction

The PingAccess Agent for Apache is an Apache module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.

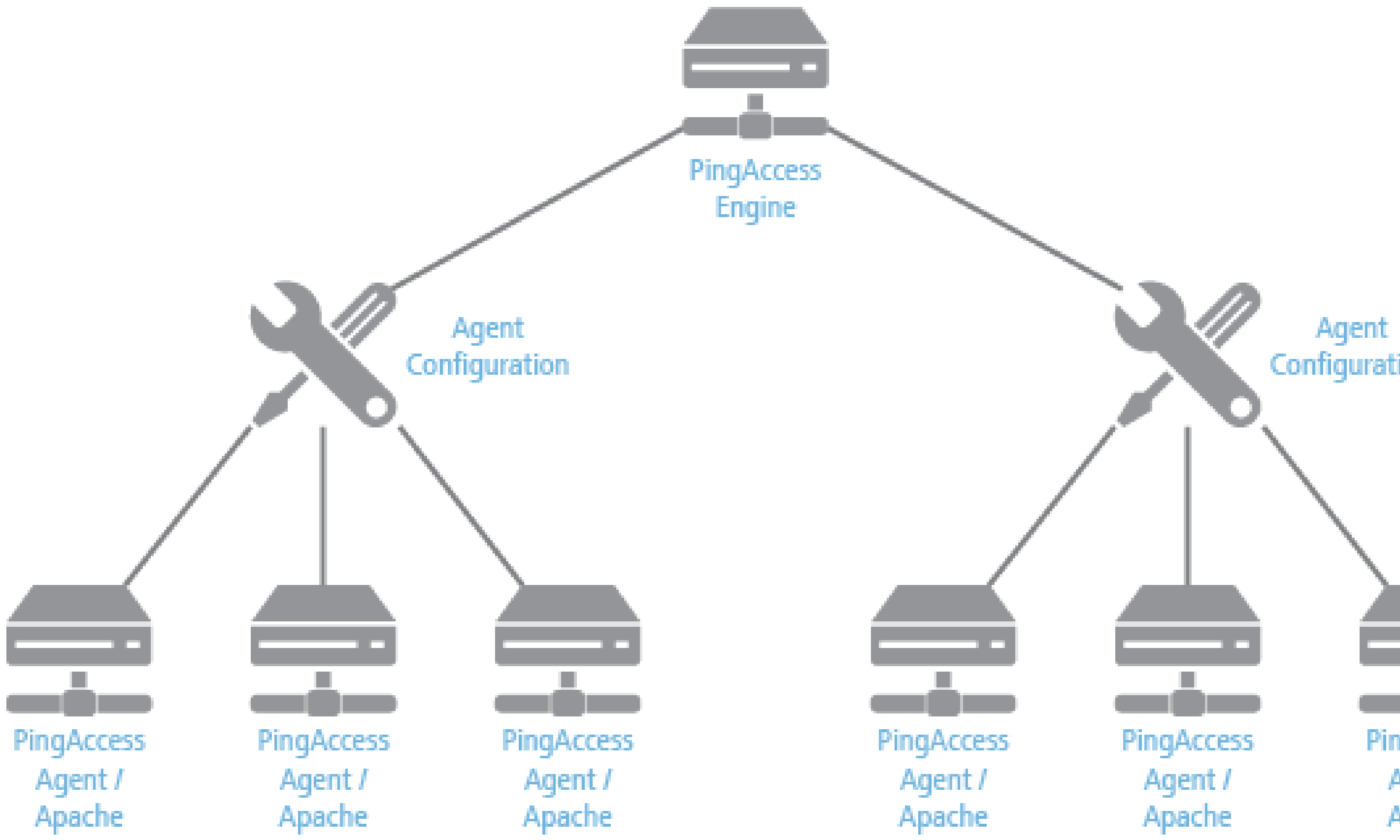


The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.

3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

Within the PingAccess Administration Console, agent nodes are configured with information that allows a PingAccess Agent to connect to the engine node to retrieve information about access control policies for resources within that agent's control. An agent configuration has a one-to-many relationship with PingAccess agents, allowing a single agent configuration bootstrap file to be used on multiple web servers within a server farm.


PingAccess Engine
Agent Configuration
Agent Configurati
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
PingAccess Agent / Apache
Pin
A
A

ⓘ **Tip:** An agent node is a shared configuration used by one or more agents, rather than a specific agent instance.

System requirements

The PingAccess Agent for Apache is supported on the following platforms:

- Apache HTTP Server 2.4 64-bit running on Microsoft Windows Server 2012 R2 Datacenter, VC14 or higher
- Apache HTTP Server 2.4 64-bit running on Microsoft Windows Server 2016, VC14 or higher

As with any system that is reachable from the Internet, the server should be properly hardened.

Installation

Before you begin

This procedure makes the following assumptions:

- The Apache configuration directory is: `C:/apache24/conf`
- The Apache modules directory is: `C:/apache24/modules`

For custom installations:

- Modify the configuration and module paths below as needed.
- Edit the included `paa.conf` to modify the values for Apache's configuration and module directories.

To install the PingAccess Agent for Apache, perform the following steps:

Steps

1. Download and extract `pingaccess-agent-apache<version>.zip`.
2. Go to the `pingaccess-agent-apache<version>` directory.
3. Copy the `paa.conf` file into the Apache configuration directory.
4. Add the following to Apache's `httpd.conf` file:

    ```
    Include conf/paa.conf
    ```

5. Copy the `paa` folder into the Apache modules directory.
6. In the PingAccess Console, go to the **Agents** page.
7. Edit a configured agent. If the agent has not yet been created, see the *PingAccess User Interface Reference Guide*.
8. In the shared secret, click **Download** to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
9. On the Agent system, create the `C:/apache24/conf.d` folder if it does not exist.
10. Copy the `<agentname>_agent.properties` file to `C:/apache24/conf.d/agent.properties`.
11. Restart Apache.

Uninstall
You can remove the PingAccess Agent from a Windows system.

Steps

1. Remove the `C:/apache24/conf.d` folder and its contents.
2. Remove the `paa` folder from the Apache modules directory.
3. Remove the following from Apache's `httpd.conf` file:

    ```
    Include conf/paa.conf
    ```

4. Remove the `paa.conf` file from the Apache configuration directory.
5. Remove the `pingaccess-agent-apache<version>` directory.
6. Remove the `pingaccess-agent-apache<version>.zip` file if it is present.
7. Restart Apache.

Configuration

The agent configuration is managed through the `paa.conf` and `agent.properties` configuration files.

The `C:/Apache24/conf/paa.conf` file contains these configuration options:

Parameter	**Definition**	**Default Value**
PaaCertificateDir	String value containing the path to the certificates extracted from the `.properties` files.	`conf.d`

Parameter	Definition	Default Value
PaaEnabled	Determines whether the agent is enabled or disabled for a specific server configuration. Valid values: `on/off` This value can be set globally; set for individual virtual hosts, directories, locations, or files; or both. The most specific value is used.	`on`

Note: If you disable the PaaEnabled parameter globally, ensure that the PaaEnabled directive is set to `on` for the PingAccess reserved application context root. This is `/pa` by default.

For example, adding this text to an included configuration file enables PingAccess for the `/pa` context root and for the `/var/www/html/one` directory:

```
<VirtualHost *:81>
    <Location /pa>
        PaaEnabled on
    </Location>
    <Directory "/var/
www/html/one">
        PaaEnabled on
    </Directory>
</VirtualHost>
```

Adding this text to an included configuration file disables PingAccess for all content in the `/var/www/html/two` directory except for files named `page2.html`.

```
<VirtualHost *:81>
    <Directory "/var/
www/html/two">
        PaaEnabled off
       <Files
 "page2.html">
            PaaEnabled
 on
       </Files>
    </Directory>
</VirtualHost>
```

Parameter	**Definition**	**Default Value**
PaaPropertyFiles	List of `.properties` files that store configuration data used to connect the agent to the PingAccess engine nodes the agent will communicate with.	`conf.d/agent.properties`
PaaEnabledNoteName	An optional parameter which defines a note name. If a request includes a note with this name and a value of `on` or `off`, this value overrides the PaaEnabled setting for that request. If you want to use this feature, you must deploy a custom module to include this note with the correct value.	`paa-enabled-note`

The configured `agent.properties` files can contain the following parameters:

Parameter	**Definition**	**Default Value**
agent.engine.configuration.scheme	The URI scheme used to connect to the engine node. Valid values are `http` and `https`.	https
agent.engine.configuration.host	The PingAccess hostname.	The value in the Agent Node's PingAccess Host field.
agent.engine.configuration.port	The port the agent connects to on the PingAccess host. This value is defined in the PingAccess `run.properties` file.	Defined in the PingAccess Admin UI
agent.engine.configuration.username	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess Admin UI
agent.engine.configuration.shared.secret	The password used to authenticate the agent to the engine.	Defined in the PingAccess Admin UI
agent.engine.configuration.bootstrap.truststore	The base64-encoded public certificate used to establish HTTPS trust by the agent to the PingAccess engine. **Note:** If you are having difficulty connecting an agent to the PingAccess engine, verify that the Agent Trusted Certificate has been configured correctly in *Agent Management*.	Generated by PingAccess

Parameter	Definition	Default Value
agent.engine.configuration.maxConnections	The number of connections a single web server worker process maintains to the PingAccess engine defined in the agent.engine.configuration.host parameter.	10
agent.engine.configuration.timeout	The maximum time (in milliseconds) a request to PingAccess can take from the agent. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.engine.configuration.connectTimeout	The maximum time (in milliseconds) the agent can take to connect to the PingAccess engine. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.cache.missInitialTimeout	The maximum time (in milliseconds) a web server worker process waits for a response to a policy cache request sent to other web server worker processes.	5
agent.cache.broker.publisherPort	The network port web server processes use to publish policy cache requests to other web server worker processes. This port is bound to the localhost network only.	3031
agent.cache.broker.subscriberPort	The network port web server processes use to receive policy cache requests from other web server worker processes. This port is bound to the localhost network only.	3032
agent.cache.maxTokens	The maximum number of tokens stored in the policy cache for a single web server worker process. A value of 0 means there is no maximum.	0

Parameter	Definition	Default Value
agent.cache.disabled	Determines whether caching of policy decisions is enabled or disabled. A value of `1` disables caching, forcing the agent to communicate with the PingAccess host any time a policy decision needs to be made. **Warning:** Disabling caching has a significant impact on the scalability of the PingAccess Policy servers, as every rule evaluation is processed by the Policy Server. This option should only be used as a last resort because of the performance penalty.	0
agent.cache.type	Controls the type of policy cache used by the agent. There are three valid values for this property: ▪ `AUTO` - The AUTO cache type determines the appropriate cache to use based on the number of worker processes. If the number of worker processes is 1, or 16 or above, the agent uses the STANDALONE cache. If the number of worker processes is between 2 and 15, the agent uses the ZMQ cache. ▪ `STANDALONE` - The STANDALONE cache type does not share policy cache entries across worker processes. ▪ `ZMQ` - The ZMQ cache type allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	AUTO

Changes to the `agent.properties` file require a restart of the web server.

Tip: See the *Performance tuning guide* for a discussion on improving agent performance.

Log configuration

The PingAccess Agent for Apache writes its information to the standard Apache error log, defined in the Apache configuration with the `ErrorLog` configuration directive. All information logged by the PingAccess Agent is prefaced with the string `[paa]`. PingAccess Agent monitoring and performance information is prefaced with the string `[paa-monitoring]`, and contains information about how long the PingAccess Agent took to fill a cache request and how long the total policy decision took.

The LogLevel used by the PingAccess Agent module is taken from the top-level `httpd.conf` configuration.

Release notes

Version history

- **Version 1.4** – July 2019

  Agent SDK for C version 1.2.0

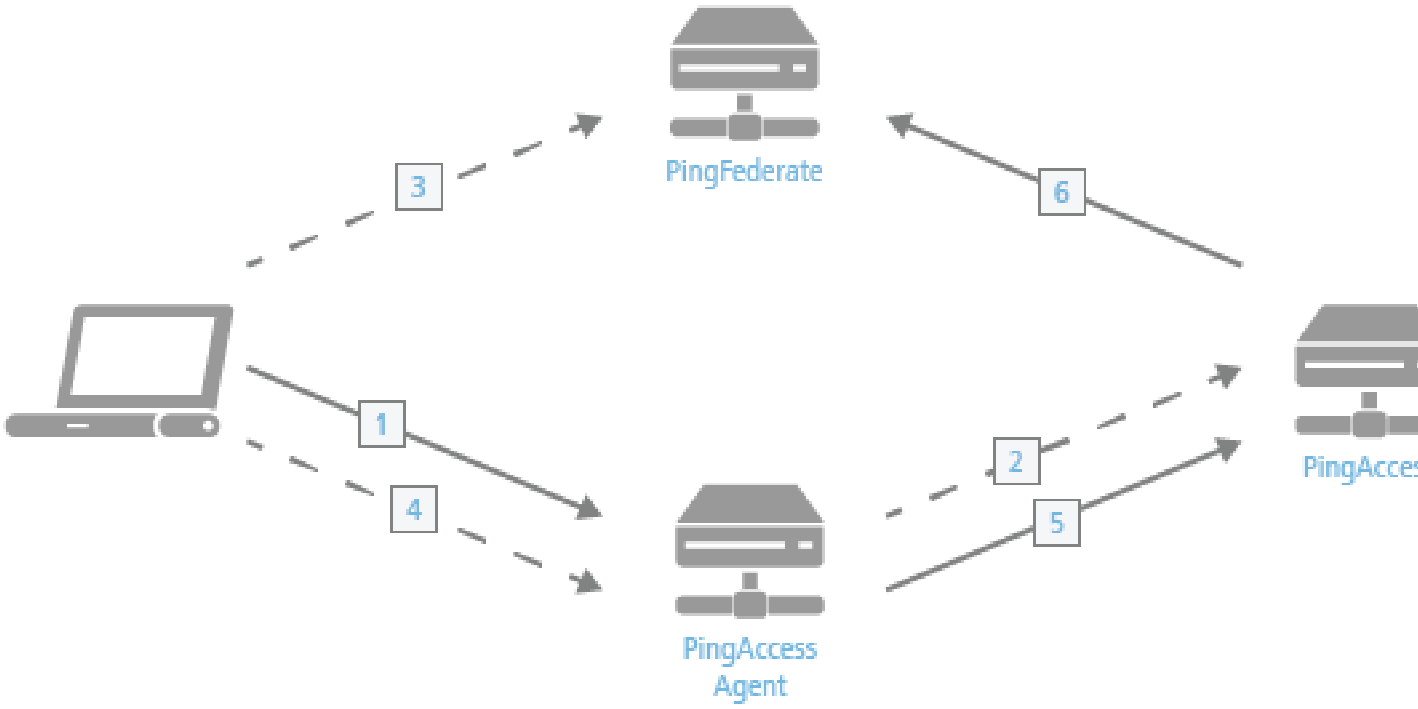
  - Initial release for Apache 2.4 on Windows

  Note: Version is aligned with PingAccess Agent for Apache (RHEL).

PingAccess Agent for IIS

Introduction

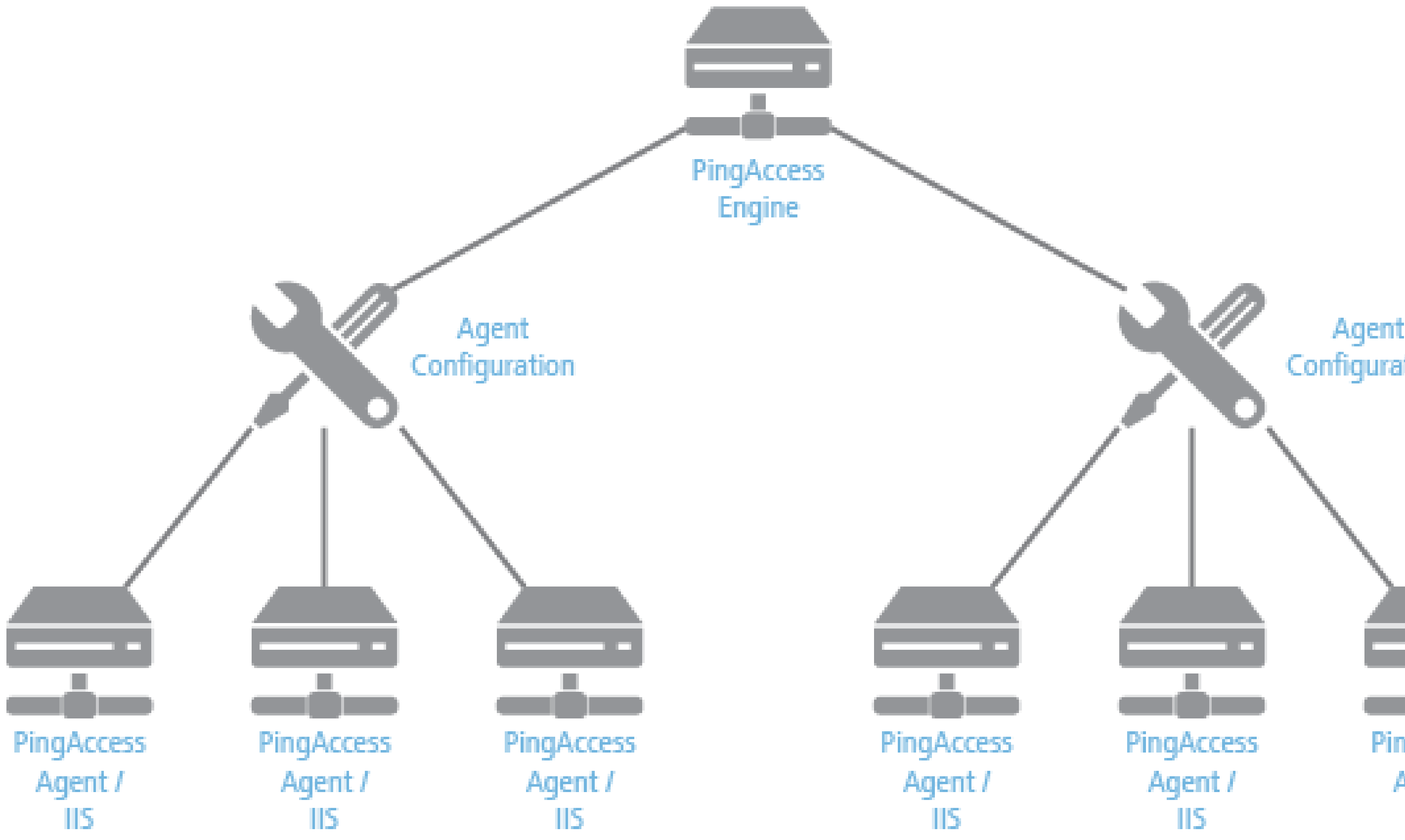
The PingAccess Agent for IIS is a Microsoft Internet Information Services module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

Within the PingAccess Administration Console, Agent Nodes are configured with information that allows a PingAccess Agent to connect to the engine node to retrieve information about access control policies for resources within that agent's control. An agent configuration has a one-to-many relationship with PingAccess agents, allowing a single agent configuration bootstrap file to be used on multiple web servers within a server farm.



ⓘ **Note:** An Agent Node is a shared configuration used by one or more agents, rather than a specific agent instance.

ⓘ **Tip:** It is possible that a problem with the PingAccess IIS agent configuration can cause all applications in an IIS application pool to become unreachable. To maximize availability of unprotected resources, place applications protected by the PingAccess agent in a separate pool.

System Requirements

To install the PingAccess Agent for IIS, the following is required:

- Microsoft Internet Information Services (IIS) 7.0 running on Windows Server 2008

- Microsoft Internet Information Services (IIS) 7.5 running on Windows Server 2008 R2
- Microsoft Internet Information Services (IIS) 8.0 running on Windows Server 2012 Datacenter Edition
- Microsoft Internet Information Services (IIS) 8.5 running on Windows Server 2012 R2 Datacenter and Standard Edition
- Microsoft Internet Information Services (IIS) 10.0 running on Windows Server 2016

Note: Only 64-bit Windows platforms and versions of IIS are supported. 32-bit compatibility mode for IIS is not supported by the PingAccess Agent for IIS.

All of the other dependencies required for the agent are included in the .msi installation package.

As with any system that is reachable from the Internet, the server should be properly hardened.

Installation

Before you begin

Prior to starting the installation of the PingAccess Agent for IIS, shut down any running programs, including the Windows Event Viewer. Running applications may cause the installation to fail.

Important: If the system is running application pools in 32-bit compatibility mode, review the *Troubleshooting* for information about preventing a known issue.

About this task

To install the PingAccess Agent for IIS, perform the following steps:

Steps

1. Unzip the PingAccess Agent for IIS zip file.

   Note: The installer cannot be run from inside the zip file; it must first be extracted.

2. Run the `pingaccess-agent-iis.msi` installer.
3. In the PingAccess Console, go to the **Agents** page.
4. Edit a configured agent. If the agent has not yet been created, follow the procedure in the *PingAccess User Interface Reference Guide*.
5. In the shared secret, click the download icon to download the configuration. The configuration file will be named `<agentname>_agent.properties`.
6. Copy the `<agentname>_agent.properties` file to `C:\Program Files\Ping Identity \PingAccess Agent for IIS\agent.properties`.
7. If the agent is running on Windows Server 2008 with IIS 7.0, perform the following steps:
   a. Download the certificate for the KeyPair associated with the Agent HttpsListener
   b. In Windows Explorer, double-click the certificate to view the certificate information.
   c. Click **Install Certificate** to start the Certificate Import Wizard, then click **Next**.
   d. When prompted to select a certificate store, select **Place all certificates in the following store**, then click **Browse**.
   e. Enable **Show physical stores**, then expand **Trusted Root Certification Authorities** and select **Local Computer**.
   f. Click **OK**, then click **Next**.
   g. Click **Finish** to exit the Wizard.

8. Restart Microsoft IIS Server on the system by performing the following steps:
    a. Launch Internet Information Services (IIS) Manager.
    b. Navigate to the web server node in the Connections tree.
    c. In the **Actions** pane, click **Restart**.

> ⓘ **Tip:** Alternatively, the command `iisreset /restart` can be used to restart the IIS service.

Uninstall
You can remove the PingAccess Agent from an IIS system.

About this task

> ⓘ **Note:** You can also remove the PingAccess Agent from an IIS system using the Add/Remove Programs option in the Windows control panel.

Steps

1. Run the `pingaccess-agent-iis.msi` installer.
   The installer displays the available workflows.
2. Select the Remove workflow.

Configuration

The PingAccess Agent for IIS configuration is managed through the Internet Information Services (IIS) Manager application. During the installation of the agent, a configuration schema extension is added to the system.webServer section. This schema extension adds the two configuration options defined in the following table:

Parameter	Definition	Default Value
PaaCertificateDir	String value containing the path to the certificates extracted from the .properties files.	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs`
PaaPropertyFiles	List of .properties files which store configuration data used to connect the agent to the PingAccess engine nodes the agent will communicate with.	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\agent.properties`

> ⓘ **Note:** It is not necessary to make any changes to these configuration parameters if the steps in the *Installation* section were followed.

The configured `agent.properties` files can contain the following parameters:

Parameter	Definition	Default Value
agent.engine.configuration.scheme	The URI scheme used to connect to the engine node. Valid values are *http* and *https*.	https
agent.engine.configuration.host	The PingAccess hostname.	The value in the Agent Node's `PingAccess Host` field.

Parameter	Definition	Default Value
agent.engine.configuration.port	The port the agent connects to on the PingAccess host. This value is defined in the PingAccess `run.properties` file.	Defined in the PingAccess Admin UI
agent.engine.configuration.username	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess Admin UI
agent.engine.configuration.checkCertRevocation	Determines whether the agent performs certificate revocation list checking against the server certificate used by the engine nodes or by a load balancer in front of the engine nodes. A value of `1` enables CRL checking, while a value of `0` disables CRL checking.	Not present by default. Treated as 1 when not specified.
agent.engine.configuration.shared.secret	The password used to authenticate the agent to the engine.	Defined in the PingAccess Admin UI
agent.engine.configuration.bootstrap.truststore	The base64-encoded public certificate used to establish HTTPS trust by the agent to the PingAccess engine. ⓘ **Note:** If you are having difficulty connecting an agent to the PingAccess engine, verify that the Agent Trusted Certificate has been configured correctly in *Agent Management*.	Generated by PingAccess
agent.engine.configuration.maxConnections	The number of connections a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host parameter`.	10
agent.engine.configuration.timeout	The maximum time (in milliseconds) a request to PingAccess can take from the agent. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.engine.configuration.connectTimeout	The maximum time (in milliseconds) the agent can take to connect to the PingAccess engine. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000

Parameter	Definition	Default Value
agent.cache.missInitialTimeout	The maximum time (in milliseconds) a web server worker process waits for a response to a policy cache request sent to other web server worker processes.	5
agent.cache.broker.publisherPort	The network port web server processes use to publish policy cache requests to other web server worker processes. This port is bound to the localhost network only.	3031
agent.cache.broker.subscriberPort	The network port web server processes use to receive policy cache requests from other web server worker processes. This port is bound to the localhost network only.	3032
agent.cache.maxTokens	The maximum number of tokens stored in the policy cache for a single web server worker process. A value of 0 means there is no maximum.	0

Parameter	**Definition**	**Default Value**
agent.cache.disabled	Determines whether caching of policy decisions is enabled or disabled. A value of `1` disables caching, forcing the agent to communicate with the PingAccess host any time a policy decision needs to be made. This option may be desired when using PingAccess 3.1 or earlier with the following rule types: ▪ Groovy Script Rule ▪ HTTP Request Rule ▪ Network Range Rule ▪ Time Range Rule **Note:** PingAccess 3.2 does require the cache be disabled in order to process these rules correctly from an agent. This may also be desirable for custom rules created using the PingAccess SDK that involve data that changes with every request within a resource and session. **Warning:** Disabling caching has a significant impact on the scalability of the PingAccess Policy servers, as every rule evaluation is processed by the Policy Server. This option should only be used as a last resort because of the performance penalty.	0

Parameter	Definition	Default Value
agent.cache.type	Controls the type of policy cache used by the agent. There are three valid values for this property: ▪ **AUTO** - The AUTO cache type determines the appropriate cache to use based on the number of worker processes. If the number of worker processes is 1, the agent uses the STANDALONE cache. If the number of worker processes is 2 or more, the agent uses the ZMQ cache. ▪ **STANDALONE** - The STANDALONE cache type does not share policy cache entries across worker processes. ▪ **ZMQ** - The ZMQ cache type allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	AUTO

Changes to the `agent.properties` file require a restart of the web server.

Tip: See the *Performance tuning guide* for discussion on improving agent performance.

Log Configuration

The PingAccess Agent for IIS installer registers the PingAccess-Agent Windows Event Log publisher when the agent is installed. This makes PingAccess Agent log information available in the Windows Event Viewer in the `Applications and Services Logs\PingAccess-Agent` folder.

PingAccess Agent for IIS logs information to one of three potential logs:

Log	Description
Admin	Contains general information messages about the module and any error messages that occur during operation. This should be a low-volume log. This log is enabled and visible by default.
Analytic	Contains monitoring, performance, and timing information. Entries in this log are useful for diagnosing performance issues. Information about the source of a policy decision for each request. This log is enabled but hidden by default.
Debug	Contains debug-level information about the module's operation. This log should only be enabled at the request of Ping Identity support technicians, as it is a very high volume log and may contain sensitive identity and token information. This log is disabled and hidden by default.

Note: To view and enable a log:

1. Click **View# Show Analytic and Debug Logs** in Event Viewer to display all of the logs.
2. Select the log that you want to enable in the console tree.

3. Click **Enable Log** in the Actions pane.

In addition to using the Windows Event Viewer, PingAccess Agent log information is accessible using the PowerShell cmdlet `get-winevent`. For example, in a PowerShell session, the content of these logs can be retrieved using the command:

```
PS> get-winevent -logname PingAccess-Agent/Admin,PingAccess-Agent/
Debug,PingAccess-Agent/Analytic -Oldest
```

Troubleshooting

The table below lists some potential problems and resolutions you may encounter with the PingAccess Agent for IIS.

Issue	Resolution
The Installer fails to successfully install the agent.	The steps listed in the *Manual Installation* procedure can be used to validate the installation and allow you to manually complete the installation. You can review the MSI installer log file for the installation to identify errors. The log file is stored in the Temp directory `C:\Users\<username>\AppData\Local\Temp` by default. The filename is not fixed, so you must locate the most recent MSI*.log file. You can direct the installer to log to a specific file by launching the installer using this command: `msiexec /l*v "<location>/paAgentInstaller.log" /i "pingaccess-agent-iis.msi"`
The Uninstall program fails to successfully remove the agent.	Follow the steps in the *Manual Removal* to remove the configuration for the PingAccess Agent for IIS.
The PingAccess-Agent/Admin log contains the error `SSL peer certificate or SSH remote key was not OK(0)`	It is likely that the hostname for the PingAccess engine being accessed does not match the hostname in the certificate used by the agent. Verify the certificate configuration, and if necessary, recreate the certificate for the Agent HTTPS Listener and recreate the Agent configuration. See *PingAccess User Interface Reference Guide* in the PingAccess documentation for more information.
500 series errors accessing protected resources	This can indicate that the PingAccess Agent failed to load, or that the Default Application Pool is stopped. Correct the issue that's causing the module load failure, and then restart the Default Application Pool. One potential cause of this is that the `agent.properties` file cannot be found or loaded. Ensure that this file is copied over as described in *Step 6* of the installation procedure.

Issue	**Resolution**

32-bit application pools crashing	This indicates that IIS attempted to load the PingAccess 64-bit agent module in an application container that is running in 32-bit mode. Modify the `applicationHost.config` file's `PingAccessAgentModule` directive in the `globalModules` section to add the following `preCondition` directive:
	`preCondition="integratedMode, bitness64"`
	For example:
	`<globalModules>`
	`<add name="PingAccessAgentModule"`
	`     image="c:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll"`
	`     preCondition="integratedMode, bitness64" />`
	`</globalModules>`

Agent does not start. Application log contains this error: `The Module name PingAccessAgentModule path (...)\paa-iis-module.dll returned an error from registration. The data is the error.`	This can indicate a corrupted or invalid `agent.properties` file. Export the `agent.properties` file from the administrative console and replace the existing file on the IIS system with the new version, as described in *Installation* on page 293.

Issue	Resolution

Agent receives an unknown protocol error when attempting to contact the administrative node	This can indicate that the operating system is using sha1 for encryption. This protocol is no longer supported by default in PingAccess.
	
	We recommend switching to sha256. If you cannot switch to sha256, you can re-enable sha1:
	
	1. Open the `run.properties` file.
	2. Add TLSv1 to the protocol list. For example:
	`tls.default.protocols=`**TLSv1,** `TLSv1.1, TLSv1.2`
	3. Add the SHA entries to the cipher suites list. For example:
	`tls.default.cipherSuites =`
	`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_RSA_WITH_AES_128_CBC_SHA256,\`
	`TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,\`
	`TLS_EMPTY_RENEGOTIATION_INFO_SCSV, \`
	**TLS_RSA_WITH_AES_128_CBC_SHA, \ TLS_DHE_RSA_WITH_AES_128_CBC_SHA, **
	**TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, **
	**TLS_ECDH_RSA_WITH_AES_128_CBC_SHA, **
	TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA

Validate IIS Configuration

About this task

For a minimal configuration of the PingAccess Agent for IIS, the following steps outline the changes made during installation that may need to be verified if the installer fails. Use this procedure as a guide for what to check if the installation did not complete successfully.

Steps

1. Stop Microsoft IIS using the following steps:
   a. Execute the command `net stop w3svc`.
   b. Execute the command `net stop was`.

2. Edit `C:\Windows\System32\inetsrv\config\applicationHost.config` and add the following line to the `sectionGroup` container with `name=system.webServer` under `configSections`:

```
<section name="paa" overrideModeDefault="Deny"
 allowDefinition="AppHostOnly" allowLocation="false" />
```

3. Add the following XML block to the `<system.webServer>` element in `C:\Windows\System32\inetsrv\config\applicationHost.config`:

```
<paa>
 <paaCertificateDir value="C:\Program Files\Ping Identity\PingAccess Agent
 for IIS\certs\" />
 <paaPropertyFiles>
  <file path="C:\Program Files\Ping Identity\PingAccess Agent for IIS
\agent.properties" />
 </paaPropertyFiles>
</paa>
```

4. Open IIS Manager and navigate to **Management** > **Configuration Editor**. Select the `system.webServer/paa` section and validate that the paths added to `applicationHost.config` are correct.
5. Register the agent module with IIS by performing the following steps:
   a. Open IIS Manager, then select the web server the agent is being added to.
   b. Click **Modules**.
   c. Click **Configure Native Modules**.
   d. Click **Register** and enter the following information:

Name	`PingAccessAgentModule`
Path	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll`

   e. Click **OK**.
   f. Click **OK**.
   g. Execute the command `iisreset /restart`.

Manual Installation

About this task

The following procedure can be used to perform a manual installation or to manually complete a partial installation if the installation failed for some reason.

ⓘ **Important:** For information about preventing a known issue on systems running application pools in 32-bit compatibility mode, see *Troubleshooting*.

ⓘ **Tip:** If you find it necessary to use this procedure due to an installation problem, please open a support ticket so the underlying issue you ran into can be addressed.

Steps

1. Stop Microsoft IIS using the following steps:
   a. Execute the command `net stop w3svc`.
   b. Execute the command `net stop was`.
2. Extract the `pingaccess-agent-iis.msi` installer file from the PA IIS Agent Distribution zip file (`pingaccess-agent-iis-x.x.x.zip`).

3. Extract the MSI installer file's contents using this command:

```
C:\Windows\System32\msiexec /a <full path to pingaccess-agent-iis.msi> /qb
 TARGETDIR=<destination path>
```

ⓘ **Note:** From this step on, this procedure will refer to the target directory as *TARGETDIR*. The files of interest are in *TARGETDIR*\PFiles.

4. Copy *TARGETDIR*\PFiles\Ping Identity\ and its contents to C:\Program Files\.
5. Download the *Microsoft Visual C++ Redistributable* and install it.
6. Add the PingAccess Agent module configuration schema to IIS by executing the following commands:
   a. `cd C:\TARGETDIR\PFiles\inetsrv\config\schema\`
   b. `copy paa_schema.xml C:\Windows\System32\inetsrv\config\schema\`
7. Edit `C:\Windows\System32\inetsrv\config\applicationHost.config` and add the following line to the `sectionGroup` container with `name=system.webServer` under `configSections`:

```
<section name="paa" overrideModeDefault="Deny"
 allowDefinition="AppHostOnly" allowLocation="false" />
```

8. Add the following XML block to the `<system.webServer>` element in `C:\Windows\System32\inetsrv\config\applicationHost.config`:

```
<paa>
 <paaCertificateDir value="C:\Program Files\Ping Identity\PingAccess Agent
 for IIS\certs\" />
 <paaPropertyFiles>
  <file path="C:\Program Files\Ping Identity\PingAccess Agent for IIS
\agent.properties" />
 </paaPropertyFiles>
</paa>
```

9. Open IIS Manager and navigate to **Management** > **Configuration Editor**. Select the `system.webServer/paa` section and validate that the paths added to `applicationHost.config` are correct.

ⓘ **Note:** If the changes are not present, ensure that you are using a 64-bit text editor. When using a 32-bit text editor, changes to this file will be transparently saved to `%SYSTEMROOT%\SysWOW64\inetsrv\applicationHost.config`.

10. Create a folder named `certs` in `C:\Program Files\Ping Identity\PingAccess Agent for IIS\`
11. Change the permissions of `C:\Program Files\Ping Identity\PingAccess Agent for IIS\certs` to include read and writer permissions for `IIS_IUSRS`. You may need to manually search for this user when modifying the permissions.
12. Register the PingAccess Agent logging publisher by performing the following steps:
    a. Execute the command:

```
C:\Windows\System32\wevtutil im paa-event-logging.xml /rf:"C:\Program
 Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll" /
mf:"C:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-
module.dll"
```

    b. Run the following three commands to ensure the logging publisher installed successfully:

```
C:\Windows\System32\wevtutil gl PingAccess-Agent/Admin
C:\Windows\System32\wevtutil gl PingAccess-Agent/Analytic
```

```
C:\Windows\System32\wevtutil gl PingAccess-Agent/Debug
```

13. Register the agent module with IIS by performing the following steps:

a. Open IIS Manager, then select the web server the agent is being added to.
b. Click **Modules**.
c. Click **Configure Native Modules**.
d. Click **Register** and enter the following information:

Name	`PingAccessAgentModule`
Path	`C:\Program Files\Ping Identity\PingAccess Agent for IIS\paa-iis-module.dll`

e. Click **OK**.
f. Click **OK**.
g. Execute the command `iisreset /restart`.

14. After IIS has restarted, use IIS Manager to ensure that the Default Application Pool has started.

> **Note:** If the Default Application Pool has not started, you will see 500 series server errors when navigating to a Site protected by the agent.

15. Continue the installation from *Step 3* of the installation procedure.

Results

The PingAccess Agent writes log information to the PingAccess-Agent logs in the Event Viewer Application and Services logs. Check these logs for any errors if the agent module does not appear to have loaded.

Manual Removal

About this task

In the event that an attempt to remove the agent from a system fails, perform the following procedure:

Steps

1. Stop Microsoft IIS using the following steps:

a. Execute the command `net stop w3svc`.
b. Execute the command `net stop was`.

2. Edit `C:\Windows\System32\inetsrv\config\applicationHost.config` and remove the following line from the `sectionGroup` container with `name=system.webServer` under `configSections`:

```
<section name="paa" overrideModeDefault="Deny"
 allowDefinition="AppHostOnly" allowLocation="false" />
```

3. Remove the following XML block from the `<system.webServer>` element in `C:\Windows\System32\inetsrv\config\applicationHost.config`:

```
<paa>
 <paaCertificateDir value="C:\Program Files\Ping Identity\PingAccess Agent
 for IIS\certs\" />
 <paaPropertyFiles>
  <file path="C:\Program Files\Ping Identity\PingAccess Agent for IIS
\agent.properties" />
 </paaPropertyFiles>
</paa>
```

4. Open IIS Manager and navigate to **Management** > **Configuration Editor**. Select the `system.webServer/paa` section and validate that the paths were properly removed from `applicationHost.config`.
5. Deregister the agent module with IIS by performing the following steps:
   a. Open IIS Manager, then select the web server the agent is being removed from
   b. Click **Modules**
   c. Click **Configure Native Modules**
   d. Select the `PingAccessAgentModule` registered module, then click **Remove**
   e. Click **OK**
   f. Execute the command `iisreset /restart`

Release Notes

Version History

- **Version 1.4** – June 2019

  Agent SDK for C version 1.2.0

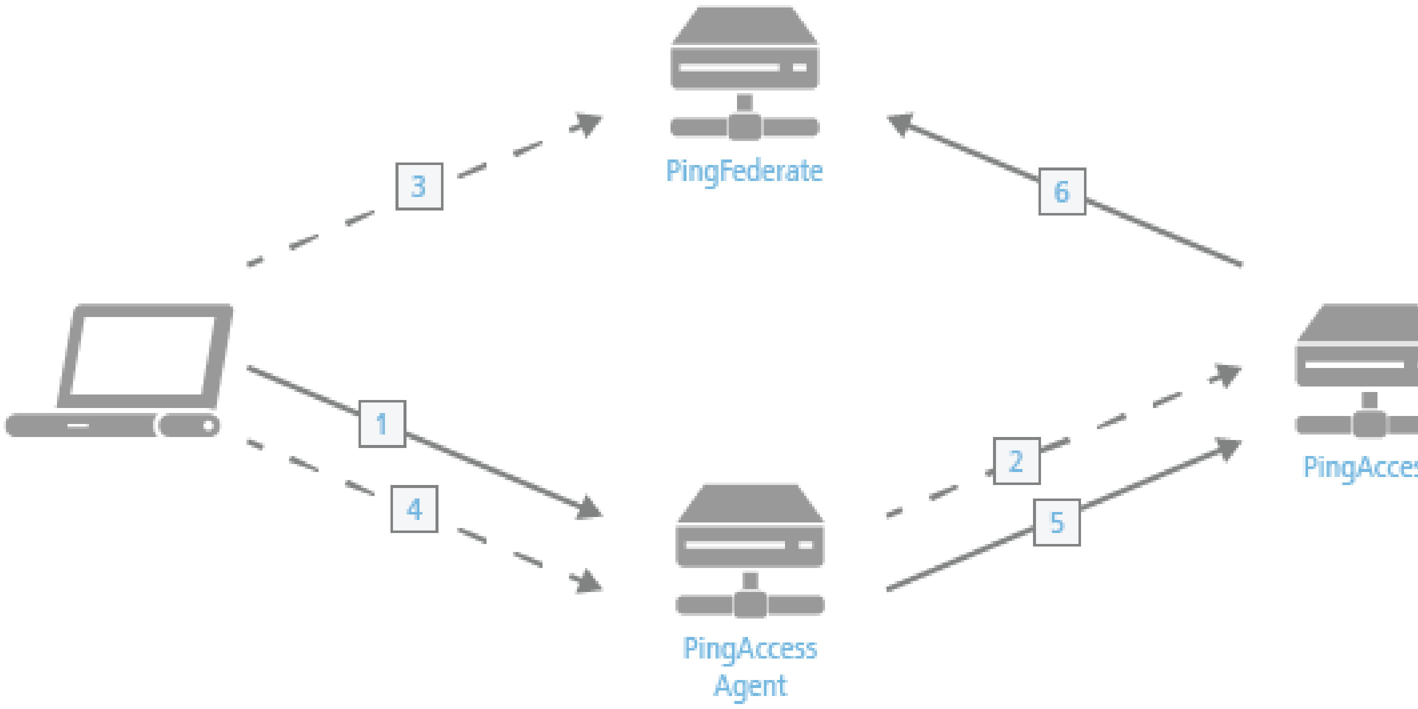
  - Added ability to set policy caching mechanism using a property in the `agent.properties` file
  - Added ability to enable or disable agent processing for a request based on a note field
  - Fixed a potential security issue
- **Version 1.3.2** – November 2018
  - Fixed a potential security issue
- **Version 1.3** – January 2017
  - Added support for IIS 10 on Windows Server 2016.
  - Updated to v1.1.1 of the PingAccess Agent SDK for C.
  - Resolved issue with IIS Preload Enabled setting.
- **Version 1.2.1** – November 2016
  - Added support for the “Preload Enabled” setting in IIS.
  - Security enhancements.
- **Version 1.2** – August 2016
  - Updated to v1.0.1 of the PingAccess Agent SDK for C.
- **Version 1.1.2** – February 2016
  - Addressed issue with custom request headers not being set when URL contains query string parameters.
- **Version 1.1.1** – September 2015
  - Addressed compatibility with the IIS plugin for WebSphere.
- **Version 1.1** – December 2014
  - Added Support for Microsoft Internet Information Services (IIS) 7.0 running on Windows Server 2008.
  - Added Support for Microsoft Internet Information Services (IIS) 7.5 running on Windows Server 2008 R2.
  - Added Support for Microsoft Internet Information Services (IIS) 8.0 running on Windows Server 2012 Datacenter Edition.
  - Corrected a potential security issue related to caching (SECBL007). This security bulletin is available in the Ping Identity Support Portal (*http://ping.force.com/Support*).
- **Version 1.0** – July 2014
  - Initial Release.

PingAccess Agent for NGINX

Introduction

Note: Download the **PingAccess Agent for NGINX** on the *PingAccess Downloads* page.

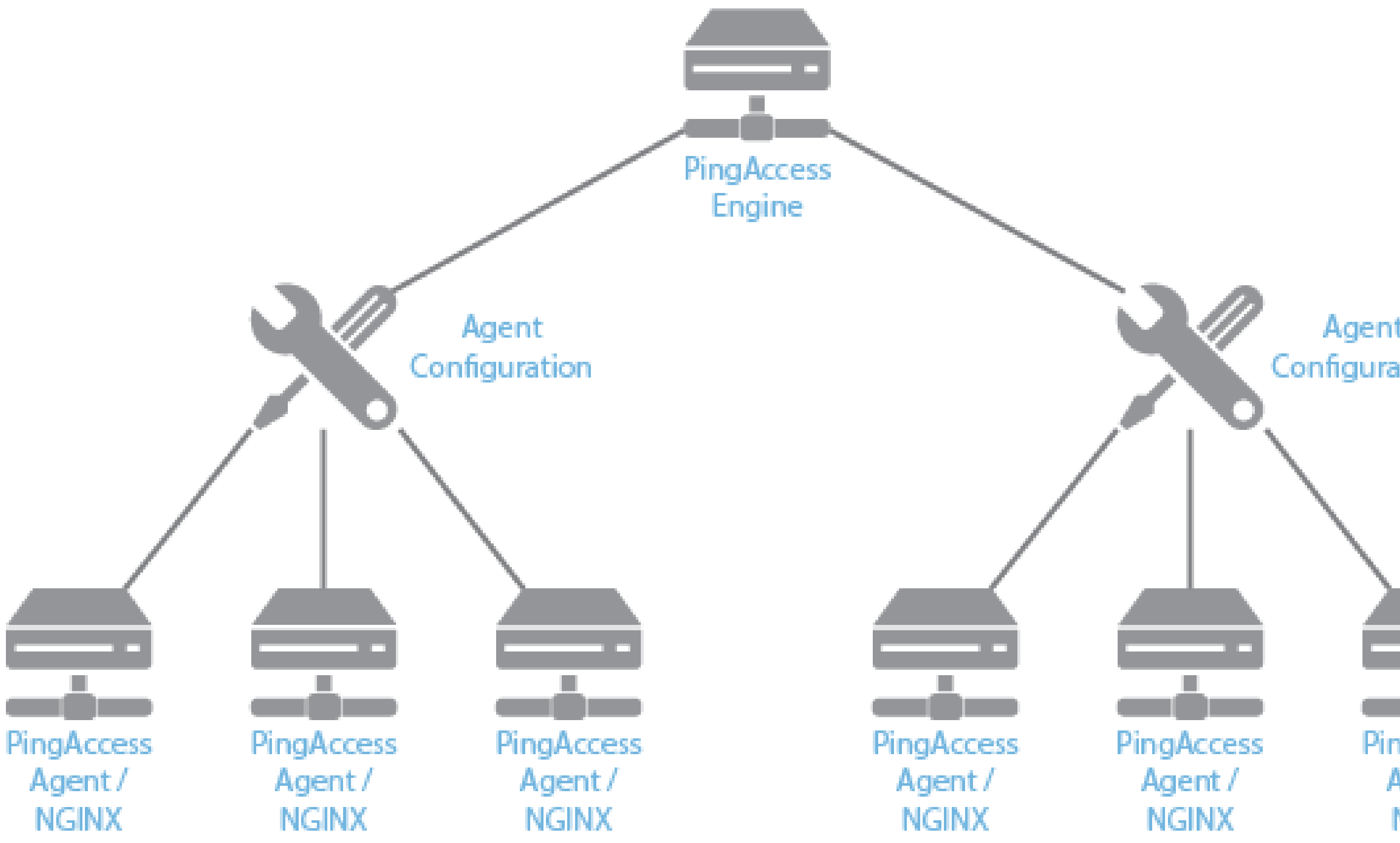
The PingAccess Agent for NGINX is an NGINX module that intercepts requests to the web server's protected resources and evaluates applicable access control policies. These policies are evaluated by either accessing a locally cached policy decision or by querying the PingAccess engine node.



The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

Within the PingAccess Administration Console, Agent Nodes are configured with information that allows a PingAccess Agent to connect to the engine node to retrieve information about access control policies for resources within that agent's control. An agent configuration has a one-to-many relationship with PingAccess agents, allowing a single agent configuration bootstrap file to be used on multiple web servers within a server farm.



ⓘ **Tip:** An Agent Node is a shared configuration used by one or more agents, rather than a specific agent instance.

System Requirements

The PingAccess Agent for NGINX is supported on the following platforms:

- Nginx Plus R15 (1.13.10) running on RedHat Enterprise Linux Server 7 (x86_64)
- Nginx Plus R16 (1.15.2) running on RedHat Enterprise Linux Server 7 (x86_64)
- Nginx Plus R17 (1.15.7) running on RedHat Enterprise Linux Server 7 (x86_64)
- Nginx Plus R18 (1.15.10) running on RedHat Enterprise Linux Server 7 (x86_64)
- Nginx Plus R19 (1.17.3) running on RedHat Enterprise Linux Server 7 (x86_64)

Installation

Before you begin

This procedure makes the following assumptions:

- The PingAccess NGINX Agent zip content is extracted to the $PINGACCESS_AGENT_NGINX folder.
- The NGINX installation is assumed to live at $NGINX. In the steps in this procedure, modify the paths specified based on where your NGINX installation and configuration files are located.
- You have downloaded the installation package from the *PingAccess Downloads* page.

About this task

To install the PingAccess Agent for NGINX, perform the following steps:

ⓘ **Note:** The Agent RPM has required dependencies that may be available via standard repositories. If these dependencies are not available in your Linux version, you can install them using the included `libpgm-5_2-0-5.2.122-32.1.x86_64.rpm`, `libsodium18-1.0.11-1.1.x86_64.rpm` and `libzmq5-4.3.1-23.6.x86_64.rpm` packages.

Steps

1. Install the NGINX module:

   ```
   yum install pingaccess-agent-nginx-*.rpm lib*.rpm
   ```

2. In the PingAccess Console, go to the **Agents** page.
3. Edit a configured agent. If the agent has not yet been created, see the **Agents** section of the *PingAccess User Interface Reference Guide*.
4. In the shared secret, click the download icon to download the agent properties file.
5. Copy the agent properties file to `$NGINX/paa/agent.properties`.
6. Add the following directive to the NGINX configuration file (`$NGINX/nginx.conf`) to load the PingAccess Agent for NGINX module:

   ```
   load_module modules/ngx_http_paa_module.so;
   ```

7. Add the following directive to the NGINX configuration file (`$NGINX/nginx.conf`) within the http {} block to configure the PingAccess Agent for NGINX module:

   ```
   include $NGINX/paa/http.conf;
   ```

   ⓘ **Important:** In PingAccess *Manage Agents*, **PingAccess Host** must match the certificate CN or Subject Alternative Name.

8. Modify the following property in the file `$NGINX/paa/http.conf` in order to enable the PingAccess Agent:

   ```
   paa_enabled on;
   ```

9. Restart the NGINX server:

   a. `sudo systemctl stop nginx`
   b. `sudo systemctl start nginx`

Uninstall

You can remove the PingAccess Agent from an NGINX system.

Steps

Run this command:

```
sudo yum remove pingaccess-agent-nginx.x86_64
```

Configuration

The PingAccess Agent for NGINX configuration is managed through the `$NGINX/paa/http.conf` configuration file.

This file contains the configuration options defined in the following table:

Parameter	Definition	Default Value
paa_properties_files	Properties file that stores configuration data used to connect the agent to the PingAccess engine nodes.	`$NGINX/paa/agent.properties`
paa_enabled on|off	Value that turns the agent on or off. This property can also be applied to server blocks within the nginx server to control which server block(s) is protected by the agent.	`off`
paa_upstream	Defines the upstream that will be used by the PingAccess Agent to route policy decision requests to PingAccess policy servers.	`pingaccess-policy-server`
paa_thread_pool	Defines the thread pool to use for blocking operations performed by the agent. Currently this only includes policy cache lookup operations when using the ZeroMQ multiprocess policy cache.	`default`

Note: It is not necessary to make any changes to http.conf if the steps in the *Installation* section were followed.

Note: Changes to the `paa_upstream` will impact how the agent communicates with PingAccess. Incorrect changes may lead to a non-functional Agent.

Note: The 'upstream pingaccess-policy-server' contains the directive 'pingaccess_servers'. This directive indicates that the servers for the containing upstream are defined by the agent.properties file. The agent only allows this directive to be specified for a single upstream.

The configured `agent.properties` files can contain the following parameters:

Parameter	Definition	Default Value
agent.engine.configuration.scheme	The URI scheme used to connect to the engine node. Valid values are *http* and *https*.	https
agent.engine.configuration.host	The PingAccess hostname.	The value in the Agent Node's `PingAccess Host` field.
agent.engine.configuration.port	The port the agent connects to on the PingAccess host. This value is defined in the PingAccess `run.properties` file.	Defined in the PingAccess Admin UI
agent.engine.configuration.username	The unique agent name that identifies the agent in PingAccess.	Defined in the PingAccess Admin UI

Parameter	Definition	Default Value
agent.engine.configuration.shared.secret	The password used to authenticate the agent to the engine.	Defined in the PingAccess Admin UI
agent.engine.configuration.bootstrap.truststore	The base64-encoded public certificate used to establish HTTPS trust by the agent to the PingAccess engine. **Note:** If you are having difficulty connecting an agent to the PingAccess engine, verify that the Agent Trusted Certificate has been configured correctly in *Agent Management*.	Generated by PingAccess
agent.engine.configuration.maxConnections	The number of connections a single web server worker process maintains to the PingAccess engine defined in the `agent.engine.configuration.host parameter`.	10
agent.engine.configuration.timeout	The maximum time (in milliseconds) a request to PingAccess can take from the agent. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.engine.configuration.connectTimeout	The maximum time (in milliseconds) the agent can take to connect to the PingAccess engine. If this time is exceeded, the client will receive a generic `500 Server Error` response.	30000
agent.cache.missInitialTimeout	The maximum time (in milliseconds) a web server worker process waits for a response to a policy cache request sent to other web server worker processes.	5
agent.cache.broker.publisherPort	The network port web server processes use to publish policy cache requests to other web server worker processes. This port is bound to the localhost network only.	3031
agent.cache.broker.subscriberPort	The network port that web server processes use to receive policy cache requests from other web server worker processes. This port is bound to the localhost network only.	3032

Parameter	Definition	Default Value
agent.cache.maxTokens	The maximum number of tokens stored in the policy cache for a single web server worker process. A value of 0 means there is no maximum.	0
agent.cache.disabled	Determines whether caching of policy decisions is enabled or disabled. A value of `1` disables caching, forcing the agent to communicate with the PingAccess host any time a policy decision needs to be made. This option may be desired when using PingAccess 3.1 or earlier with the following rule types: ▪ Groovy Script Rule ▪ HTTP Request Rule ▪ Network Range Rule ▪ Time Range Rule PingAccess 3.2 and later does not require the cache be disabled in order to process these rules correctly from an agent. **Warning:** Disabling caching has a significant impact on the scalability of the PingAccess Policy servers, as every rule evaluation is processed by the Policy Server. This option should only be used as a last resort because of the performance penalty.	0

Parameter	**Definition**	**Default Value**
agent.engine.configuration.failover.hosts	The **Hostname** and **Port** of the PingAccess server where the agent should send requests in the event of a failover from the PingAccess Host. **Note:** If this parameter is set, the upstream block name in `$NGINX/paa/http.conf` needs to be modified to a name that will be found in the certificate associated with the PingAccess Agent HTTPS Listener. For example, if your PingAccess certificate contains name '`pa.nginx`', set the upstream name to `upstream pa.nginx`.	Defined in the PingAccess Admin UI
agent.engine.configuration.failover.failedRetryTimeout	Seconds before retrying a failed PingAccess server.	60
agent.engine.configuration.failover.MaxRetries	The maximum number of retries before considering a PingAccess server unavailable.	2
agent.cache.type	Controls the type of policy cache used by the agent. There are three valid values for this property: ▪ **AUTO** - The AUTO cache type determines the appropriate cache to use based on the number of worker processes. If the number of worker processes is 1, the agent uses the STANDALONE cache. If the number of worker processes is 2 or more, the agent uses the ZMQ cache. ▪ **STANDALONE** - The STANDALONE cache type does not share policy cache entries across worker processes. ▪ **ZMQ** - The ZMQ cache type allows the agent to share policy cache entries across all worker processes using ZeroMQ for inter-process communication.	AUTO

Changes to the `agent.properties` file require a restart of the web server.

ⓘ **Tip:** See **Agent Tuning** in the PingAccess *Performance tuning reference guide* for a discussion on improving agent performance.

Release Notes

Version History

- **Version 2.0.1** – June 2019

  Agent SDK for C version 1.2.0

  - Fixed a potential security issue
- **Version 2.0** – February 2019
  - The PingAccess Agent for NGINX now leverages the built-in, event-driven HTTP stack in NGINX to communicate with PingAccess policy servers. Previously, the agent used its own HTTP client (implemented with libcurl) to communicate with PingAccess policy servers. In certain cases, this architecture lead to poor scalability. By using NGINX's built-in, event-driven HTTP stack, the agent is able to achieve superior scalability over previous versions.
  - Fixed a potential security issue
- **Version 1.1.1** – July 2017
  - Support for starting and stopping NGINX via the systemctl command
  - Resolved issue with SSL connectivity
- **Version 1.1** – March 2017
  - Updates to meet NGINX certification requirements
- **Version 1.0** – January 2017
  - Initial Release

PingAccess Agent Protocol

PAAP Overview

The PingAccess Agent Protocol (PAAP) is an HTTP-based protocol for communication and interaction between PingAccess (PA) and PingAccess agents.

An agent typically sits in front of a web application or other protected resource on the web server or load balancer (e.g.: Apache or Microsoft IIS).

PAAP is HTTP-based and utilizes a few custom status codes and headers. One goal of basing the protocol on HTTP is to enable an agent, which runs in an HTTP environment, to utilize concepts and code/libraries already at its disposal to do its job.

The majority of the responsibilities reside within PA. The intent of this protocol is to make the agent a relatively “dumb” agent, largely shielded from the configuration and processing details, and allowing for policies to be maintained centrally in PA. This means that agents do not need to know about the signing and encryption keys used by PA or PingFederate. By following this model, the protocol allows agents and PA to be versioned and upgraded independently of one another.

The protocol described here is supported by PingAccess version 3.0 and higher.

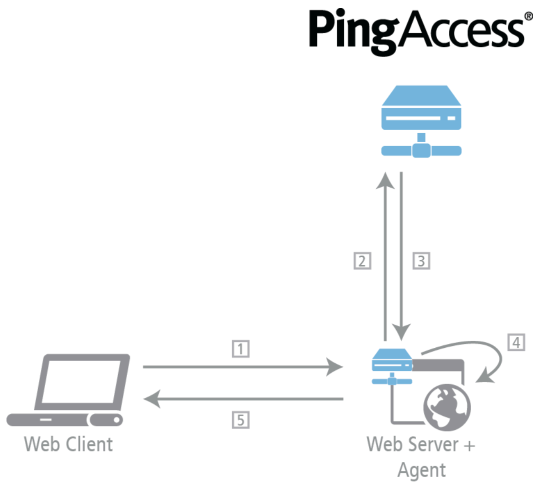
ⓘ **Note:**

The prefix “vnd-pi-” was chosen for the PAAP protocol headers defined in this document. In this context, “vnd” indicates a vendor extension, and “pi” represents Ping Identity. Custom status codes were selected after consulting the *Hypertext Transfer Protocol (HTTP) Status Code Registry*, with the intention of avoiding any conflicts.

PingAccess agent protocol flow

The PingAccess agent protocol has a set flow by which requests from clients are evalulated and managed.

The PingAccess agent protocol starts with the client (a web browser, OAuth client, or any type of HTTP client) making a request for an application resource. The agent sits in front of the resource and intercepts the request. To determine what to do with it, the agent forwards a portion of the request to PA. The response from PA instructs the agent whether to allow the original request, as well as any additional actions that should be taken prior to handing it off to the application. It also includes instructions for actions to be performed before sending the corresponding response.



The protocol flow goes through these steps:

1. Client request
2. Agent request
3. Agent response
4. Modified client request
5. Client response

PAAP client request

The PingAccess agent protocol flow begins when the client makes an HTTP 1.1 request to a server where an agent is set up in the filter or interception chain in front of an application or other protected resource.

Unauthenticated user request

This request is coming from an unauthenticated user.

```
GET /application/headers HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
```

```
Cookie: nonce=6424266c-ca9b-4e1f-9fde-d1860bfa2582
```

OIDC Connect flow request

This request is part of the OpenID Connect flow for authentication. The POST body is not shown for brevity.

```
POST /pa/oidc/cb HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Content-Length: 1557
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Origin: https://rhel-test.englab.corp.pingidentity.com:9031
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a
```

Authenticated resource request

This request is an authenticated request for a resource.

```
GET /application/headers HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie:
 PA.post=eyJraWQiOiJhcCIsImFsZyI6IkVTMjU2In0.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Z
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw
```

PAAP agent request

When the agent intercepts a client request, it makes a correlated request to PA to determine if the request is authorized and what action to take on it.

The agent request is made by the agent after it receives a client request to determine what actions to take. PA returns the agent response to the agent communicating the action. The agent request effectively mirrors the client request, except for the differences described below.

The Request-Line of the agent request is identical to the Request-Line of the client request. Unless otherwise specified below, all headers in the agent request are sent as they appear in the client request.

The message-body of the client request (if any) is omitted from the initial agent request. If PA needs the body, an HTTP 477 response, as defined in the agent response section, is returned. PA will be able to service the vast majority of agent requests without having to see the message body. The body from the initial agent request is omitted because PA will not need it to make a policy decision in most instances, and removing it provides an opportunity for significant performance efficiencies. The HTTP 477 response mechanism provides PA a way to get the body in the less common cases where it is needed.

HTTP request headers

The following HTTP request headers may require additional agent processing:

Content-Length

The Content-Length header in the agent request will have a value matching the message-body of the request being sent. For the initial agent request, which is sent without the message-body, the Content-Length will be zero. A subsequent agent request resulting from a 477 will send a Content-Length that indicates the size of the entity-body of the request, which is the same as in the client request.

vnd-pi-expect

This header allows the agent to communicate its needs to PA. “!477” is the only defined value, which tells PA that the agent request contains all the data the agent it is capable of sending regarding the client request. PA should never respond with a 477 to an agent request that has “!477” as the value of the `vnd-pi-expect` request header. Note that while *the expect header from RFC 2616* with an expectation-extension could convey the same semantics, Ping Identity elected not to use it due to language in the RFC that suggests intermediaries might/should/must reject a request using an expectation-extension they don’t understand with a 417.

A simple and effective approach for an agent implementation is to send the initial agent request with no body, a content-length of zero, and omitting the vnd-pi-expect header. The header `vnd-pi-expect: !477` is only ever sent when an agent receives a 477 response to its initial request. In other words, the initial agent request never has a vnd-pi-expect header, while a second agent request in response to an HTTP 477 response always has `!477` as the value for the vnd-pi-expect header.

PA should never respond to a GET request with a 477, but following this standard allows an agent to handle such an occurrence in an appropriate way.

vnd-pi-v

Indicates the version of PAAP the agent is using. The value is “1.0”.

vnd-pi-authz

This header is similar to the *authorization header defined in RFC 2616* but is specifically intended to enable an agent to authenticate itself to PA. The syntax for the “credentials” value of the header is the same as the *section 2.1 of The OAuth 2.0 Authorization Framework: Bearer Token Usage* .

The header looks like this:

```
vnd-pi- authz: Bearer <token>
```

Where `<token>` is a secret shared between PA and the PAA.

In some cases, unrestricted access to the agent protocol at PA might create an information leakage vulnerability. The custom headers returned in the agent response, for example, may reveal internal details of applications or infrastructure that needs to be protected. Potentially worse, the values might reveal content from encrypted WAM tokens or reference access tokens. Authenticating PAAs to PA is one means of mitigating the concern.

Authentication is optional; when it is required is at the discretion of PA. Authentication of the agent to PA is intended to be a static deployment option and, as such, no challenge response constructs are defined. Failed authentication – missing credentials when required or invalid credentials – is indicative of either a configuration problem or unauthorized access attempt. In such circumstances, PA responds with an HTTP 403 and should include the vnd-pi-authz header in the response using a quoted string value with human readable information to help troubleshoot and allow for differentiation from an unauthorized end-user. An agent may send the 403 response or a 500 response to the client, depending on which is most appropriate.

vnd-pi-resource-cache

Indicates that for the given host the vnd-pi-resource-cache and vnd-pi-resource-cache-ttl headers, defined in the caching part of the next section, are to be returned in the agent response. Generally an agent will include this header when it needs to first establish its resource definition cache for a particular host, or when its current cache is stale or invalid. When an agent request includes the vnd-pi-resource-cache header, the agent response should include the vnd-pi-resource-cache and vnd-pi-resource-cache-ttl headers. An agent must be prepared to handle an agent response that omits those headers. The literal value “requested” can be used by the agent to request the resource cache data. For example:

```
vnd-pi-resource-cache: requested
```

X-Forwarded-For

The X-Forwarded-For header contains the originating IP address of a client making a request. If no X-Forwarded-For header is present in the client request, it is added to the agent request with a value indicating the IP address of the client making the connection. If an X-Forwarded-For header is present in the client request, the IP address of the client is added to the end of the delimited list of IP addresses this header contains when sent in the agent request.

Host and X-Forwarded-Host

The agent sets the Host header in the agent request as it appeared in the client request, unless it is unable to easily manipulate the Host header. In the event that it could not modify the Host header, the X-Forwarded-Host header contains the original host requested by the client. If X-Forwarded-Host is already present in the client request, it is sent along unchanged in the agent request.

X-Forwarded-Proto

If X-Forwarded-Proto is present in the client request, it is sent along unchanged in the agent request. Otherwise, if the scheme used in the client request is different than the agent request (https vs. http), set the X-Forwarded-Proto header in the agent request to the scheme used in the client request. This header can be omitted if the client request and the agent request use the same scheme.

PA determines the requested resource and constructs self-referential URIs using the contents of the request, including the headers listed above. The scheme is determined from the client’s connection and the X-Forwarded-Proto header, with the latter taking precedence when present. The host and port are determined from the Host and X-Forwarded-Host headers, with the latter taking precedence when present.

Policy decision request

This request is a policy decision request for an unauthenticated user:

```
GET /application/headers HTTP/1.1
Host: http://example.com/
```

```
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: nonce=6424266c-ca9b-4e1f-9fde-d1860bfa2582
vnd-pi-v: 1.0
X-Forwarded-For: 172.30.3.248
vnd-pi-resource-cache: requested
X-Forwarded-Proto: http
vnd-pi-authz: Bearer Agent:htZ2W39EfAPLQd8w9cRT6y
```

Agent request without POST body

This request is an agent request (in this case, the OpenID Connect callback in a web session POST login type) without the POST body included:

```
POST /pa/oidc/cb HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Content-Length: 0
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Origin: https://rhel-test.englab.corp.pingidentity.com:9031
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a vnd-pi-v: 1.0
X-Forwarded-For: 172.30.3.248
vnd-pi-resource-cache: requested
X-Forwarded-Proto: http
vnd-pi-authz:Bearer Agent:htZ2W39EfAPLQd8w9cRT6y
```

Agent request with POST body

This request is a policy decision request with the POST body included. Note the vnd-pi-expect: !477 header that disallows the HTTP 477 agent response to request the body (as it is already included):

```
POST /pa/oidc/cb HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Content-Length: 1557
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
Origin: https://rhel-test.englab.corp.pingidentity.com:9031
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip,deflate
Accept-Language: en-US,en;q=0.8
Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a
vnd-pi-v: 1.0
X-Forwarded-For: 172.30.3.248
vnd-pi-resource-cache: requested
X-Forwarded-Proto: http
vnd-pi-expect: !477
vnd-pi-authz: Bearer Agent:htZ2W39EfAPLQd8w9cRT6y
```

Authenticated user request

This request is for a resource by an authenticated user:

```
GET /application/headers HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie:
 PA.post=eyJraWQiOiJhcCIsImFsZyI6IkVTMjU2In0.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Z
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw
vnd-pi-v: 1.0
X-Forwarded-For: 172.30.3.248
vnd-pi-resource-cache: requested
X-Forwarded-Proto: http
vnd-pi-authz: Bearer Agent:htZ2W39EfAPLQd8w9cRT6y
```

PAAP agent response

When PA receives an agent request, it sends an agent response that includes an authorization decision as well as any additional actions for the agent to perform on the client request, or requests additional information from the agent.

Any HTTP status code other than those listed below indicates that the client request was not permitted. In that case, the content of the agent response (including the status-line, the message-body, and all headers not named by the header defined below) is sent back to the client as the content of the client response. This lets PA direct the client (when applicable) to PF for user authentication via redirection or even auto-post form. This also lets PA communicate error conditions and negative authorization decisions to the client with a consistent look and feel.

Custom HTTP status codes

The following custom status codes indicate that the client request is allowed or that additional information is needed in order to make a policy decision.

HTTP 477 Request Body Required

A 477 response indicates that request requires the request body to make a policy decision. The agent should repeat the agent request and include the request body. The subsequent response will include the policy decision and any actions to be taken.

PA does not need the message body to respond to the vast majority of agent requests. This status code allows for optimization of the protocol by not sending unnecessary data by default while providing a way to ask for it when needed. The only case where PA currently requires the body is to evaluate an auto-posted OpenID Connect Authentication Response. This enables the agent to not have to know about or configure any callback redirect_uri locations, but rather push that off to PA. The agent only knows it received a POST request and that PA asked for the message-body in order to process it.

After the agent repeats the agent request with the request body, PA responds to the agent with a generic HTTP response code. Because the only time a 477 is returned by PA is to get the an auto-posted OpenID Connect Authentication Response at the redirect URI, PA never returns a 277 after a 477.

PA should never respond to a GET request with a 477.

HTTP 277 Allowed

The 277 response indicates the client request is authorized and should be allowed to continue. Additional actions to perform on the client request and corresponding client response are indicated by one or more of the headers listed in the HTTP 277 headers section.

HTTP 277 headers

These headers can be included in the agent response if the response code HTTP 277 was used.

vnd-pi-set-req-headers

The value of this header is a comma-delimited list of header names from the agent response to be included as headers for the client request. If any included headers already exist in the client request, the values are overwritten with the values from the corresponding agent response header. If a header is named which is not present in the agent response, it is removed from the client request.

This allows, for example, PA to make user attribute information available to the protected application in the headers, and to guard against header injection by the client. In order to guard against malicious header injection by end users, use this mechanism to expose user data to the application. PA should include all header names used in a given context, even if they have no value, so headers supplied by the client using the same names will be ignored.

vnd-pi-append-req-headers

The value of this header is a comma-delimited list of header names from the agent response to add to the client request headers. Any existing named headers already present in the client request are not overwritten, but new headers are added with the values from the agent response. If a header is named which is not present in the agent response, no action is taken for that header name in the client request.

vnd-pi-set-req-vars

The value of this header is a comma-delimited list of header names from the agent response that are to be set in the client request as request scoped variables or attributes in an appropriate manner for the environment in which the agent resides. Examples might include environment variables in Apache or request attributes in a servlet container.

vnd-pi-sub

This header is used to identify the header that contains the subject or username for the transaction. This typically will be a header also named in vnd-pi-set-req-headers, as those generally expose user information to the backend application via headers in the modified client request. It may name a header not in the vnd-pi-set-req-headers list, if the agent or environment in which it's deployed needs to know the username/subject in a way that differs from header injection to the client request. The vnd-pi-sub header should name a single-valued header, but if it names one with multiple values, the agent should use only one. If vnd-pi-sub names a header which is not present in the agent response, it should be ignored by the agent.

vnd-pi-set-resp-headers

The value of this header is a comma-delimited list of header names from the agent response which are to be set as headers of the client response. If any of the named headers already exist in the client response, the values are overwritten with the values of those headers from the agent response. If a header is named which is not present in the agent response, it is removed from the client response.

vnd-pi-append-resp-headers

The value of this header is a comma-delimited list of header names from the agent response which are to be added to the headers of the client response. Any existing named headers already present in the client response are not overwritten, but new headers are added with the values from the agent response. If a header is named which is not present in the agent response, no action is taken for that header name in the client response.

This allows, for example, PA to set or reset the PA WAM token as a cookie in the user's browser.

In general, for any particular request or response header name, PA should only indicate either a set or an append directive. However, with an agent response, the set directive takes precedence. One way the agent might accomplish that is by applying all append operations first, followed by the set operations.

Common headers

These headers can be used in an agent response of any status code.

vnd-pi-omit-resp-headers

The value of this header is a comma-delimited list of header names from the agent response to omit from the headers in the client response. When present, this list implicitly includes the vnd-pi-omit-resp-headers header.

Caching

A number of caching directives are aimed at reducing calls from the agent to PA and improving performance.

Resource Definition Caching

When the vnd-pi-resource-cache request header is present in the agent request, PA includes the following headers in the agent response. This enables the agent to make initial decisions on handling of client requests without having to consult PA directly.

vnd-pi-resource-cache

This is a multi-valued header and, in keeping with *Section 4.2 of RFC 2616*, the values may be comma-delimited, or multiple message-header fields with the same name may be included. The order of the values is significant and, in servicing future requests based on the cache, an agent should evaluate them in the order they were received.

Each value represents a group of resources and some directives about how the agent should handle requests for URIs within that group of resources. The values are made up of multiple parts delimited by semicolons.

- **path** – The path part defines the path(s) against which requests are matched. The value is one or more space-delimited quoted strings. Each quoted string is a path value, which may contain wildcards using the asterisk (*) character. So, for example, `path="/app/*"` would match any requested path that starts with `/app/`. While `path="/app1/*" "/app2/*"` would match anything under `/app1/` or `/app2/`. Similarly, `path="*.jpg" "*.gif" "*.png"` would match anything ending with those common image file extensions. If no wildcard is present, the values must exactly match.
- **cs** – Indicates if the values in the path are case-sensitive. Valid values are Y and N. If this component is omitted, the default value is Y.
- **method** – Indicates the method or methods against which requests are matched. The value is one or more space-delimited method names (i.e. GET, POST, PUT, etc). If this component isomitted, all methods are allowed and should match for the resource.
- **kind** – Indicates the kind of resource and the general level of access control protection which is to be applied to it - i.e. whether access to the resource (and related resources as per the path values) requires an authorization token. Valid values are P, U or C. The value P (meaning 'Protected') says that a token is required for access and the token-type and token-name indicate the token of interest so that the token value can be used in caching the response and response headers to service future requests. The value U (meaning 'Unprotected') indicates that no token is necessary for access and that any future request (within the cache time-to -live) will be allowed. The value C (meaning 'Consult' with PA) means that the agent must always make an agent request to PA in order to service the client request.
- **token-type** – The token-type part indicates what kind of token was used in making the authorization decision and what token type to use in making future cache queries. Its value is either C or A. A value of C indicates that a cookie was used to make the access control decision, and that future requests with the same cookie value for the cookie named in the token-name part can use the cached content. A value of A indicates that an authorization header was used to make the access control decision, and that future requests with the same credentials for the authorization scheme named by the token-name directive can use the cached content.
- **token-name** – The token-type says what type of token for which to cache specific user details for a particular token. However, there may be more than one token for a particular type in a request. The token-name value disambiguates that situation by specifying which one to use. The token-name value is either the name of a cookie (for WAM) or the name of an authorization scheme (the "Bearer" value for OAuth), when token-type value is C or A, respectively.

> **Note:** When this value is a cookie, the cookie name is case-sensitive, as implied by *RFC 6265*. When this value is the name of an authorization scheme, per *section 1.2 of RFC 2617*, the value is not case-sensitive. When using the token-name header, ensure that the value follows the appropriate case-sensitivity requirements.

The resource-cache list is valid for and scoped to the host in the client request. When building the resource-cache, PA includes resources associated with the virtual host that matches the host of the request as well as wildcard virtual host resources

vnd-pi-resource-cache-ttl

The value of the resource cache time-to -live header is an integer indicating the number of seconds from the time the response was sent that the values of the vnd-pi-resource-cache header could be cached and used. For example, the following header instructs the agent to cache the resource definitions for the next ten minutes:

```
vnd-pi-resource-cache-ttl: 600
```

The agent can use the vnd-pi-resource-cache header in an agent request to ask PA for new vnd-pi-resource-cache and vnd-pi-resource-cache-ttl values when the time-to-live on its current resource cache has, or is about to, elapse.

PA provides configurability over the resource cache time-to-live value to balance performance and security goals.

An example vnd-pi-resource-cache response header is shown below. This example tells the client that all requests with a path starting with `/pa/oidc/` are to have the agent make an agent request to PA to determine what to do. Next, it tells PA that requests with a jpg, gif or png suffix are allowed to pass through. Requests for a path that starts with `/canada/` require a PA WAM token which will be a cookie named PA.cad. Requests for a path that starts with `/usa/` require a PA WAM token which will be a cookie named PA.usd. All other requests, indicated by the slash wildcard path are allowed. Note that the request path is matched against the paths defined in the resource-cache in order from top to bottom. The vnd-pi-resource-cache-ttl tells the agent to use the resource cache for the next hour.

```
vnd-pi-resource-cache: path="/pa/oidc/*"; kind=C
vnd-pi-resource-cache: path="/*.jpg" "*.gif" "*.png";
 method=GET; kind=U
vnd-pi-resource-cache: path="/canada/*"; cs=N; kind=P; token-
type=C; token-name=PA.cad
vnd-pi-resource-cache: path="/usa/*"; kind=P; token-type=C;
 token-name=PA.usd
vnd-pi-resource-cache: path="/*" ; kind=U
vnd-pi-resource-cache-ttl: 3600
```

Note that the above is semantically equivalent to the following headers where the multiple vnd-pi-resource-cache header fields are combined into one.

```
vnd-pi-resource-cache: path="/pa/oidc/*"; kind=C, path="/*.jpg"
 "*.gif" "*.png"; method=GET; kind=U, path="/canada/*"; cs=N;
 kind=P; token-type=C; token-name=PA.cad, path="/usa/*"; kind=P;
 token-type=C; token-name=PA.usd, path="/*" ; kind=U
vnd-pi-resource-cache-ttl: 3600
```

Individual token and agent response caching

The resource-cache defined in the previous section gives the agent meta-information about caching data for request handling. This section describes how, in some cases, data from an individual agent response can be cached relative to a particular token and used to service future client requests with the same token so that PA doesn't need to be called on every client request.

The resource-cache defined in the previous section gives the agent directives about how to handle requests based on host, method and path. The agent iterates its resource-cache list in order until it finds a match based on those values. Additional caching can be done for particular kinds of resources as follows.

When the "kind" of the resource-cache is P, a token (as indicated by the token-type and token-name) is required but previous agent responses for a token can be cached for efficiency. If the agent does not have a cached agent response for a particular token value, it must make an agent request to PA in order to determine how to handle the client request. The data from that agent response can then be cached using the value of the indicated token as a key. An empty, null, or missing token should also be considered a valid cache key in order to support the ‘anonymous’ access use case, where a WAM token is not necessary for access but, if such a token is available, user attributes from it should be exposed to the application. An agent response to a request that does not have the indicated token-type/token-name will likely contain a vnd-pi-set-req-headers directive that names non-existent headers in order to ensure they are stripped from the modified client request. This prevents injection of those header values by the client, even in an ‘anonymous’ case.

When caching individual agent responses relative to particular tokens, the protocol directives state that the token value is obtained from the client request. However, there is one important special case where, for efficiency, the token value can be obtained from the agent response. That special case is for resources with a “kind” of P and token-type of C that receive a 277 agent response containing a positive vnd-pi-token-cache-ttl header value and avnd-pi-append-resp-headers that includes the set-cookie header. Under those conditions, the agent can examine the set-cookie headers for a cookie name matching the token-name of the resource and use the value of that cookie as the token value to cache the agent response. Note that the agent should also exclude that set-cookie header from the cached agent response content. This allows the cache to be established for an individual token in only one agent request to PA when the token in the cookie is updated and set on the client.

Though the token relative caching is primarily intended as an optimization to store and reuse data associated with status code 277 responses, the cache header defined below is valid on any agent response, and agents should be prepared to cache all agent responses, rather than just 277 responses.

In general, individual agent responses for resources of kind of C are not cached. The one exception is the special case of a 477 response code where an agent can cache the 477 (which tells it to send the request body on the initial agent request along with a !477 value for the vnd-pi-expect header) for a specific request URI, until the vnd-pi-resource-cache-ttl passes.

A kind value of U indicates that no agent request or response is necessary for the client request, so no additional caching is necessary.

vnd-pi-token-cache-ttl

Indicates the number of seconds from the time the agent response is issued that it can be cached relative to a specific token value. The agent must make a new agent request if the TTL on the cache entry of an individual token has expired, or if no cache entry exists.

The TTL should correlate to the life of the token itself - i.e. the time-to-live must be shorter than the expiration, and it needs to also allow for updates to the inactivity timeout within a reasonable threshold.

There are many tradeoffs involved, thus PA enables tuning and configuration options for the TTL directive.

In the event that the token is empty, null , or missing (i.e., the “anonymous” use case), the value of vnd-pi-token-cache-ttl can be the same as the value of the vnd-pi-resource-cache-ttl.

Early Cache Invalidation

PA may include the following header in an agent response to instruct the agent to invalidate its cache. The agent may need to do this , for example, as a result of configuration changes.

vnd-pi-cache-invalidated

The value of this header is a numeric value representing the number of seconds from 1970-01-01T0:0:0Z UTC (the epoch) until the UTC date/time of the cache invalidation event. The value is an indicator of the most recent event that would trigger an invalidation of the cache associated with the host (or X-Forwarded-Host) of the agent request correlated to the agent response in which this header appears. An agent may ignore an invalidation directive for a timestamp that it has already processed.

It is difficult for PA to know the cache state of any particular agent or group of agents. So it is not reasonable to expect PA to identify the exact responses that should include the cache invalidation directive. Use of the timestamp as the header value allows PA to send the vnd-pi-cache-invalidated more indiscriminately while allowing the agent to relatively easily determine if it needs to, or already has, taken action with respect to a specific invalidation event.

Change Propagation and Caching

An agent populates and expunges its cache over time. As a result, configuration changes in PA may take some time to propagate, and may yield a mixed set of old and new behavior.

The invalidation directive set via the vnd-pi-cache-invalidated header on the agent response is intended to provide some help seeing changes take effect inside the TTL window. Though caching does reduce the number of calls made from an agent to PA, there are still many requests that will necessitate the call and allow the vnd-pi-cache-invalidated header to be sent to an agent.

OpenID Connect authentication

This response is passed through to the client to begin the OpenID Connect authentication process. The status and headers are passed directly through to the client.

```
HTTP/1.1 302 Found
Date: Wed, 17 Sep 2014 23:10:30 GMT
Content-Length: 0
Location: https://rhel-test.englab.corp.pingidentity.com:9031/
as/authorization.oauth2?response_type=x_post
%20id_token&client_id=pa_wam&redirect_uri=http://example.com/pa/
oidc/
cb&state=aHR0cDovL3JoZWw2NS9hcHBsaWNhdGlvbi9oZWFkZXJzIEFwcGxpY2F0aW9uIFJvb3QrUmVzb
gs0lMY&scope=openid%20profile%20address%20email%20phone
Set-Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a; Path=/;
 HttpOnly
vnd-pi-resource-cache: path="/pa/*";kind=C,path="/
application/*" "/application";cs=Y;kind=P;token-
type=C;token-name=PA.post,path="/protected/*" "/
protected";cs=Y;kind=P;token-type=C;token-name=PA.post,path="/
httpbin/headers*";cs=Y;kind=P;token-type=C;token-
name=PA.post,path="/httpbin/*" "/httpbin";cs=Y;kind=P;token-
type=C;token-name=PA.post
vnd-pi-resource-cache-ttl: 900
vnd-pi-token-cache-ttl: 300
```

Request for POST body

This response requests the POST body that was omitted from the initial agent request:

```
HTTP/1.1 477 Request Body Required
Date: Wed, 17 Sep 2014 23:10:35 GMT
Content-Length: 0
vnd-pi-resource-cache: path="/pa/*";kind=C,path="/
application/*" "/application";cs=Y;kind=P;token-
type=C;token-name=PA.post,path="/protected/*" "/
protected";cs=Y;kind=P;token-type=C;token-name=PA.post,path="/
httpbin/headers*";cs=Y;kind=P;token-type=C;token-
name=PA.post,path="/httpbin/*" "/httpbin";cs=Y;kind=P;token-
type=C;token-name=PA.post
```

```
vnd-pi-resource-cache-ttl: 900
```

Redirect

This response issues a redirect. The status code and headers are passed directly through to the client.

```
HTTP/1.1 302 Found
Date: Wed, 17 Sep 2014 23:10:36 GMT
Content-Length: 0
Location: http://example.com/application/headers
Set-Cookie:
 PA.post=eyJraWQiOiJhcCIsImFsZyI6IkVTMjU2In0.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw;
 Path=/; HttpOnly
Set-Cookie: nonce=; Path=/; Expires=Thu, 01-Jan-1970 00:00:00
 GMT
vnd-pi-resource-cache: path="/pa/*";kind=C,path="/
application/*" "/application";cs=Y;kind=P;token-
type=C;token-name=PA.post,path="/protected/*" "/
protected";cs=Y;kind=P;token-type=C;token-name=PA.post,path="/
httpbin/headers*";cs=Y;kind=P;token-type=C;token-
name=PA.post,path="/httpbin/*" "/httpbin";cs=Y;kind=P;token-
type=C;token-name=PA.post
vnd-pi-resource-cache-ttl: 900
```

Grant access

This response grants access and allows the client request through to the application with the appropriate application headers set and with the caching directives.

```
HTTP/1.1 277 Allowed
Date: Wed, 17 Sep 2014 23:10:36 GMT
Content-Length: 0
vnd-pi-resource-cache: path="/pa/*";kind=C,path="/
application/*" "/application";cs=Y;kind=P;token-
type=C;token-name=PA.post,path="/protected/*" "/
protected";cs=Y;kind=P;token-type=C;token-name=PA.post,path="/
httpbin/headers*";cs=Y;kind=P;token-type=C;token-
name=PA.post,path="/httpbin/*" "/httpbin";cs=Y;kind=P;token-
type=C;token-name=PA.post
vnd-pi-resource-cache-ttl: 900
vnd-pi-token-cache-ttl: 300
USER: joe
vnd-pi-sub: USER
vnd-pi-set-req-headers: USER
```

PAAP modified client request

If the agent response status is 277, the client request is modified according to the directives in the agent response and the request is passed along to the application or allowed to continue processing in the HTTP processing pipeline of the environment in which the agent is deployed.

Additional HTTP headers

This example shows the additional HTTP headers added as specified by PA.

```
GET /application/headers HTTP/1.1
Host: http://example.com/
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/
xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120
 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie:
 PA.post=eyJraWQiOiJhcCIsImFsZyI6IkVTMjU2In0.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zb
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw
USER: joe
```

PAAP client response

The client response is the HTTP response corresponding to the client request.

If the agent response status code is anything other than 277, its content is sent back to the client as the content of the client response, minus any headers identified for exclusion. If the agent response status code is 277, the client request is passed to the protected application, and the client response is the response from the application with any header modifications indicated by the agent response.

Pass-through from agent response

This client response shows the direct pass-through of the status code and headers from the agent response.

See the agent response OpenID Connect authentication example for a related example.

```
HTTP/1.1 302 Found
Date: Wed, 17 Sep 2014 23:10:30 GMT
Content-Length: 0 Location: https://rhel-
test.englab.corp.pingidentity.com:9031/as/
authorization.oauth2?response_type=x_post
%20id_token&client_id=pa_wam&redirect_uri=http://example.com/pa/
oidc/
cb&state=aHR0cDovL3JoZWw2NS9hcHBsaWNhdGlvbi9oZWFkZXJzIEFwcGxpY2F0aW9uIFJvb3QrUmVzb
gs0lMY&scope=openid%20profile%20address%20email%20phone
Set-Cookie: nonce=b000c6a2-4a03-4bde-be29-956456cd1d2a; Path=/;
 HttpOnly
```

Client response with redirect

This client response shows the direct pass-through of the status code and headers from the agent response.

See the agent response redirect example for a related example.

```
HTTP/1.1 302 Found
Date: Wed, 17 Sep 2014 23:10:36 GMT
Content-Length: 0
Location: http://example.com/application/headers
Set-Cookie:
 PA.post=eyJraWQiOiJhcCIsImFsZyI6IkVTMjU2In0.eyJ6b25laW5mbyI6IkFtZXJpY2FcL05ld19Zk
PingAccess 3.x SpecificationAgent
 Protocol Specification V1.0Page 25
 JmZW1hbGUiLCJwcm9maWxlIjoiaHR0cHM6XC9cL3d3dy5waW5naWRlbnRpdHkuY29tXC9wcm9kdWN0c1w
j0BDdLoGeVdqWD35n9ZxFhphEHFe7tfQ6onKAjRdXLR5rtwPBkJHkLaTLD8Yqcsf0izVw;
 Path=/; HttpOnly
Set-Cookie: nonce=; Path=/; Expires=Thu, 01-Jan-1970 00:00:00
 GMT
```

PingAccess Agent SDK for C

Preface

This documentation provides technical guidance for using the PingAccess Agent SDK for C. Developers can use this guide along with the API documentation for the SDK and sample source code to implement custom agents that use the PingAccess Agent Protocol to integrate with a PingAccess policy server.

Intended Audience

This guide is intended for application developers and system administrators responsible for implementing a C-based PingAccess agent. The reader should be familiar with C software development principles and practices. It describes the use of the SDK within a sample Agent for Apache.

Additional documentation

The SDK documentation provides detailed reference information for developers. After unzipping the `pingaccess-agent-c-sdk-`*`version`*`.zip` package, the API documentation can be accessed with a web browser by viewing the file *`AGENT_SDK_C_HOME`*`/apidocs/index.html`. The current version of the API documentation may also be found online at *https://www.pingidentity.com/content/dam/developer/documentation/pingaccess/agent-c-sdk/1-1-4/apidocs/index.html*

Introduction

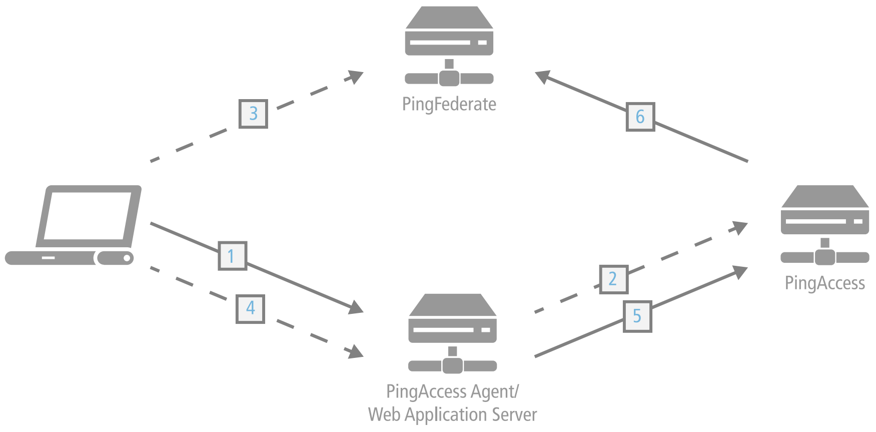
The PingAccess Agent SDK for C provides an API and sample code to enable developers to build agents for C or C++-based application and web servers.

Supported platforms

- RedHat Enterprise Linux Server 6 (32 bit)
- RedHat Enterprise Linux Server 6 (64 bit)
- RedHat Enterprise Linux Server 7 (32 bit)
- RedHat Enterprise Linux Server 7 (64 bit)
- SUSE Linux Enterprise Server 11 SP4 (64 bit)

- SUSE Linux Enterprise Server 12 SP2 (64 bit)

The PingAccess Agent SDK for C provides an API and sample code to enable developers to build agents for C-based application and web servers. Agents provide access management features to their containing server by relying on central PingAccess servers over the PingAccess Agent Protocol. The *PingAccess Agent Protocol Specification* is available from the Ping Identity support portal.



The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

The PingAccess Agent SDK for C consists of the following components:

SDK (C Agent)

The SDK is a set of C header files that represent the interface to the library that implements the PingAccess Agent Protocol.

C Agent libraries

The C libraries implement the PingAccess Agent Protocol. There are binaries for Red Hat Enterprise Linux 6/7 as well as for Windows.

PingAccess Agent SDK for C API documentation

Each of the interfaces defined in the header files is fully documented.

Apache Agent Sample

`<AGENT_SDK_FOR_C_HOME>/sample` : The Apache Agent Sample demonstrates how the SDK integrates into Apache as an Apache module that is integrated with the Apache request processing workflow. The provided source code and module configuration provide a functional example for how to integrate the SDK into an existing web application. The sample can be modified in-place and recompiled using **`make`** to test customizations to the Sample code for your environment.

Note: This sample code demonstrates how to implement the PingAccess Agent as an Apache module and has been qualified in the following environments:

- Red Hat Enterprise Linux 6 (RHEL6), 64-bit

- Red Hat Enterprise Linux 7 (RHEL7), 64-bit

The Apache Agent itself is production-quality and can be used either as-is or as a starting point for further development. While Ping Identity provides this as a sample, the only versions that are fully supported in production are the precompiled versions available from the Ping Identity download site.

The sample includes instructions for how to configure the sample as a PingAccess Agent to protect websites within its scope. Note that further hardening of the Apache server configuration or of the sample configuration file may be required.

If you need assistance using the PingAccess Agent SDK for C, visit the Ping Identity *Support Center* (ping.force.com/Support) to see how we can help you with your application. You may also engage the Ping Identity Professional Services team for assistance with developing customizations.

Getting Started with the PingAccess Agent SDK for C

Agent SDK for C directory structure

The PingAccess Agent SDK for C directory contains the following subdirectories:

`/`

This directory contains the Agent SDK for C `README.md`, which contains information developers will need in order to develop agents using the SDK. It also contains `ReadMeFirst.pdf` and `Legal.pdf`, which contain general information about the kit and third-party licenses used by components of the SDK.

`/apidocs`

API documentation for the SDK. Open `index.html` to access the API documentation content.

`/include`

Agent SDK header files.

`/lib`

32-bit and 64-bit libraries for Red Hat Enterprise Linux 6/7 and Windows, including third-party dependencies required by the SDK.

`/sample`

Sample source code for an agent for Apache. This sample agent uses the SDK, and includes a sample configuration file for Apache to use the sample agent to enforce authentication and access control policies.

Agent SDK for C sample code

The Agent SDK for C sample code is available both in the SDK distribution and on github at *https://github.com/pingidentity/pa-agent-c-sdk-sample-apache*.

Before building the sample code, ensure you have the PingAccess Agent SDK for C archive, the GNU **`make`** utility and associated compiler utilities installed with your compiler, and Apache and its development libraries.

The sample uses Apache and assumes that the PingAccess Agent SDK for C can be referenced as a dependency. For more details about specific dependencies and requirements, as well as instructions on how to build the sample code, see `*AGENT_SDK_C_HOME*/sample/readme.md`.

Release notes

- **Version 1.2 - June 2019**
  - Fixed a potential security issue.
- **Version 1.1.5 - February 2019**
  - Added support for FreeBSD 8
- **Version 1.1.4 - October 2018**
  - Fixed potential security issues.
- **Version 1.1.3 - August 2018**
  - Updated version of libcurl to fix an issue where libcurl was only checking the first SAN in the server certificate.
  - Fixed a potential security issue.
- **Version 1.1.2 - March 2017**
  - Added support for:
    - SUSE Linux Enterprise Server 11 SP4 (x86_64)
    - SUSE Linux Enterprise Server 12 SP2 (x86_64)
- **Version 1.1.1 - January 2017**
  - Established a workaround for a *known issue* in the Network Security Services library that results in a memory leak when the agent closes a HTTPS connection to a PingAccess policy server. For more information, see *this KB article*.
  - Fixed an issue where duplicate headers were being included in the backend request to the PingAccess Engine causing the request for content to be blocked by the agent.
- **Version 1.1 - November 2016**
  - Added policy server failover support. Policy server failover support is only provided by the SDK when using the libcurl HTTP client.
- **Version 1.0.2 - September 2016**
  - Fixed an issue where agents could not communicate with PingAccess servers using a certificate that is signed by a certificate authority because the CRL Distribution Point extension is missing. This issue is limited to agents on Windows deployments.
  - Addressed a potential security vulnerability. This issue is limited to Windows deployments.
- **Version 1.0.1 - May 2016**
  - Fixed an issue with ZeroMQ policy cache where a terminated process could cause a condition that resulted in unexpected CPU utilization.
- **Version 1.0 - April 2016**
  - Initial Release.

PingAccess Agent SDK for Java

Preface

This document provides technical guidance for using the PingAccess Agent SDK for Java. Developers can use this guide along with the Javadocs for the Java Agent API and sample source code to implement the PingAccess Agent Protocol in custom agents.

Intended audience

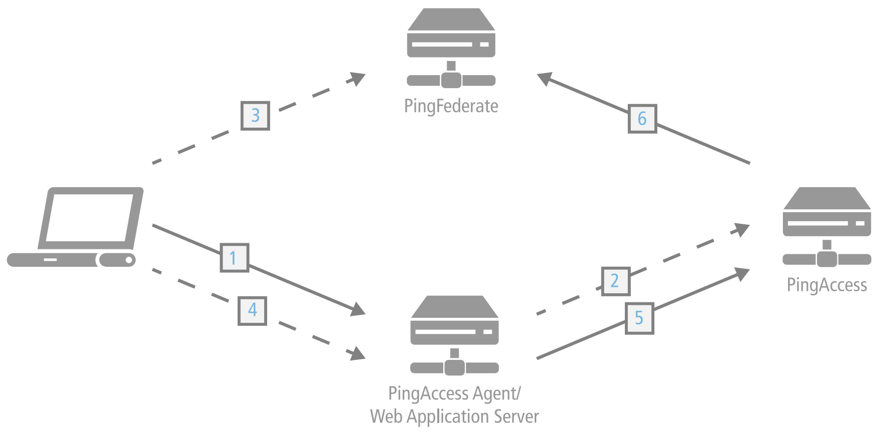
This guide is intended for application developers and system administrators responsible for implementing a Java PingAccess Agent. The reader should be familiar with Java software-development principles and practices. It describes the use of the SDK within a sample Java Servlet Filter.

Additional documentation

The Java Agent API Javadocs provide detailed reference information for developers. After unzipping the `pingaccess-agent-java-sdk-1.1.3.zip` package, the Javadocs can be accessed with a web browser by viewing the file *`<AGENT_SDK_JAVA_HOME>`*`/apidocs/index.html`.

Introduction

The PingAccess Agent SDK for Java provides an API and sample code to enable developers to build agents for Java-based application and web servers. Agents provide access management features to their containing server by relying on central PingAccess servers over the PingAccess Agent Protocol. The *PingAccess Agent Protocol Specification* is available from the Ping Identity support portal.



The process used when a PingAccess Agent is added to the policy decision process is as follows:

1. The client accesses a resource. If the user is already authenticated, this process continues with step 5.
2. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then redirects the client to PingFederate to establish a session.
3. The user logs in, and PingFederate creates the session.
4. The client is then redirected back to the resource.
5. The agent asks PingAccess for instructions. PingAccess checks the URL policy and determines that it is a protected resource. PingAccess then checks the session token and determines that it is valid.
6. If session revocation is enabled, PingAccess checks and updates the central session revocation list. If the session is valid, the agent is instructed to set identity HTTP headers.

The PingAccess Agent SDK for Java consists of the following components:

Java Agent API (Java Agent)

`pingaccess-agent-java-api-1.1.3.0.jar`: The Java Agent API is a set of classes that implement the PingAccess Agent Protocol.

PingAccess Agent SDK for Java

`pingaccess-agent-java-sdk-1.1.3.zip`: The PingAccess Agent SDK for Java package.

Servlet Filter Sample

`<AGENT_SDK_FOR_JAVA_HOME>``/sample` : The Servlet Filter Sample demonstrates how the Java Agent API integrates into a Java Servlet container. The provided source code, logging configuration and deployment descriptor provide a functional example for how to integrate the Java Agent API into

an existing web application. The sample can be modified in place and recompiled using Maven to test customizations to the Servlet Filter Sample code for your environment.

> **Note:** This sample code demonstrates how to implement a servlet filter and has been qualified on Apache Tomcat 7. The filter itself is production quality and can be used either as-is or as a starting point for further development. Application configuration within the sample demonstrates how to associate the filter with a servlet (namely, in `web.xml`). Further hardening of this file or the application server configuration may be required.

If you need assistance using the PingAccess Agent SDK for Java, visit the Ping Identity *Support Center* (ping.force.com/Support) to see how we can help you with your application. You may also engage the Ping Identity Global Client Services team for assistance with developing customizations.

Agent SDK directory structure

The PingAccess Agent SDK for Java directory (pingaccess-agent-java-sdk-1.1.3) contains the following:

`/apidocs`

The Javadocs for the Java Agent API. Open `index.html` in this directory to access the Javadocs content.

`/dist`

The directory containing `pingaccess-agent-java-api-1.1.3.0.jar`

`/sample`

A directory containing `src` and `target` directories for building a Java Servlet Filter. This filter uses the Java Agent API, an `agent.properties` configuration exported from PingAccess, and the `init-params` from the web application `web.xml` file to enforce resource policy decisions configured in PingAccess.

Agent SDK prerequisites

Before you start, ensure you have the Java SDK, *Apache Maven* (maven.apache.org) and an application server (e.g. Apache Tomcat) installed. The sample uses Apache Maven and assumes that the Java Agent API can be referenced as a dependency. It references Ping Identity's public Maven repository, located at:

```
http://maven.pingidentity.com/release
```

If Internet access is unavailable, there are two other ways to reference the Java Agent API. First, once Apache Maven is installed, install the Java Agent API into your local dependency repository by executing the following command:

```
mvn install:install-file -Dfile=<AGENT_SDK_JAVA_HOME>/dist/pingaccess-agent-
java-api-1.1.3.0.jar -DgroupId=com.pingidentity -DartifactId=pingaccess-
agent-java-api -Dversion=1.1.3.0 -Dpackaging=jar
```

Alternatively, update the dependency in your `pom.xml` to point to the local installation:

```
<dependency>
        <groupId>com.pingidentity</groupId>
        <artifactId>pingaccess-agent-java-api</artifactId>
        <version>1.1.3.0</version>
        <scope>system</scope>
        <systemPath><AGENT_SDK_JAVA_HOME>/dist/pingaccess-agent-java-
api-1.1.3.0.jar</systemPath>
</dependency>
```

With either of these options, replace <*AGENT_SDK_JAVA_HOME*> with the absolute path to the unzipped `pingaccess-agent-java-sdk-1.1.3.0` directory.

How to install the servlet filter sample

Before you begin

Ensure you have the PingAccess Agent SDK for Java, Apache Maven, and Apache Tomcat. These instructions assume that you are using Apache Tomcat.

About this task

- The servlet filter sample is installed under `<AGENT_SDK_JAVA_HOME>/sample`.
- A deployed version of the servlet filter is under `<AGENT_SDK_JAVA_HOME>/sample/target/agent-sample`.

For the initial setup of the web application, we assume you already have Tomcat or another application server set up on the same machine hosting PingAccess. Out of the box, PingAccess generates self-signed server certificates for listeners servicing runtime ports with the hostname `localhost`. By default, the servlet filter sample configures the Java Agent (Java Agent API) to use "strict" certificate checking for communications with PingAccess. The Java Agent will not be able to communicate with PingAccess over HTTPS if it is not also on `localhost` because of strict hostname checking. If PingAccess already has a server certificate configured with a valid hostname other than `localhost`, then you can deploy the Java Agent into a container on another system.

If you cannot setup the application server on the same system as an existing PingAccess service, and that PingAccess deployment still uses the default `localhost` server certificate for the Agent port, there is another option. You can change the default `strict` certificate checking in `agent-sample/WEB-INF/web.xml` to `test`. Please see the comments in `agent-sample/WEB-INF/web.xml` for more detail.

The agent-sample (servlet filter sample) web application is meant to demonstrate the features of the Java Agent within the context of a functional, standalone sample application. The servlet filter sample uses the Java Agent to intercept requests bound for sample servlet and will accept or reject them based on the configured PingAccess policy. The sample servlet only prints out headers, cookies, and other parameters it receives in the request.

Steps

1. In the Tomcat `webapps` directory, create a directory called `ROOT`.
2. Copy the `WEB-INF`, `META-INF`, and `assets` contents from `/sample/target/agent-sample/` into `webapps/ROOT`.

   This sample servlet filter must run as `/` to properly carry out the OpenID Connect workflow.
3. In the Tomcat `bin` directory, create a script called `setenv.sh` (Linux) or `setenv.bat` (Windows) with the following contents:

   - For Linux:

     ```
     export CATALINA_OPTS="-Dlog4j.configurationFile=<PATH_TO_TOMCAT_ROOT>/
     webapps/ROOT/WEB-INF/logs/log4j2.xml -
     Dserver.log.file=<PATH_TO_TOMCAT_ROOT>/webapps/ROOT/WEB-INF/logs/
     server.log"
     ```
   - For Windows:

     ```
     set CATALINA_OPTS=="-Dlog4j.configurationFile=<PATH_TO_TOMCAT_ROOT>/
     webapps/ROOT/WEB-INF/logs/log4j2.xml -
     ```

```
Dserver.log.file=<PATH_TO_TOMCAT_ROOT>/webapps/ROOT/WEB-INF/logs/
server.log"
```

The Agent servlet filter logging is configured in `webapps/ROOT/WEB-INF/logs/log4j2.xml`, and outputs to `webapps/ROOT/WEB-INF/logs/server.log`

4. If running Tomcat on Linux, execute the command `chmod a+x setenv.sh` to make this script executable.
5. Configure a PingAccess Agent.
6. Configure an Application and associate the new Agent with it.
7. When configuring an Agent through the PingAccess administration console, it automatically exports the agent properties file. Copy the downloaded properties file to `webapps/ROOT/WEB-INF/agent-config/agent.properties`.

Important: If Tomcat is running on Java version 7, some version 8 cipher suites are unavailable. This may lead to errors.

To work around this issue, edit `agent.properties` to remove the following cipher suites from `agent.ssl.ciphers`:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

8. Start Tomcat
9. Start a browser and navigate to `http://<HOST>:<PORT>/sample`

   The values for *<HOST>* and *<PORT>* here need to match the Tomcat configuration in use.

Note: If your Tomcat server is not set up to use HTTPS, ensure that any related Web Sessions do not have the **Secure** option enabled.

PingAccess Agent SDK for Java Release History

Version 1.1.3 - July 2018

Resolved issues

- Fixed an issue where site cookies were not being set properly for HTTP redirects in the servlet container environment (e.g., Tomcat).
- Fixed an issue where an error was being generated by the SDK sample implementation of `ClientHttpServletRequest.getHeaders` when calling a header that does not exist.

Version 1.1.2 - June 2017

Resolved issues

- Fixed an issue where the Java agent was handling the PingAccess `set-cookie` header incorrectly.
- Fixed an issue where the Java agent wasn't correctly processing multiple `set-cookie` headers sent by PingAccess.
- Fixed an issue where the SDK sample implementation was not correctly enforcing the PingAccess Agent Protocol directives when the ClientHttpServletRequest `getCookies` method was called. This resulted in a discrepancy between the cookie request headers returned from the `getHeader*` methods and the `getCookies` method.

Version 1.1.1 - April 2017

Resolved issues

- Fixed an issue where unknown attributes should be ignored
- Fixed an issue where percent-encoded sequences in resource paths were being handled incorrectly
- Fixed an issue where an "Index out of bounds" exception occurs if a cookie value is "".

Version 1.1 - August 2015

Resolved issues

- Fixed an issue where OAuth API response headers were getting trimmed
- Fixed an issue where the Java Agent enforced the requirement of a username and shared secret
- Fixed an issue where the Agent was not handling a 477 response correctly

Version 1.0 - June 2015

- Initial Release

PingAccess Addon SDK for Java

Preface

This document provides technical guidance for using the PingAccess Add-on SDK. Developers can use this guide, in conjunction with the installed Javadocs, to extend the functionality of the PingAccess server.

> **Important:** A restart of PingAccess is required after the deployment of any custom plugins written in Java.

Intended audience

This guide is intended for application developers and system administrators responsible for extending PingAccess. The reader should be familiar with Java software-development principles and practices. It describes the development of:

- SiteAuthenticators
- Rules
- Identity mappings
- Load balancing strategies
- Locale override service

Additional documentation

- The PingAccess Javadocs provide detailed reference information for developers. The Javadocs can be accessed with a web browser by viewing the file `PA_HOME/sdk/apidocs/index.html`.

Introduction

The PingAccess Add-on SDK provides extension points that let users customize certain behaviors of PingAccess to suit their needs. This SDK provides the means to develop, compile, and deploy custom extensions to PingAccess.

The PingAccess Add-on SDK provides the following extension points:

RuleInterceptor

An interface for developing custom Rule implementations to control authorization logic in policies.

SiteAuthenticatorInterceptor

An interface for developing custom Site Authenticators to control how PingAccess (operating as a proxy) is able to integrate with web servers or services it is protecting.

IdentityMappingPlugin

An interface for developing custom Identity Mappings to provide user identity information to an Application within PingAccess.

LoadBalancingPlugin

An interface for developing custom Load Balancing strategies that provide the logic for load balancing requests to Target Hosts configured for a Site.

LocaleOverrideService

An interface for developing custom logic for resolving the locale of a request used for localization.

If you need assistance using the SDK, visit the Ping Identity *Support Center* (ping.force.com/Support) to see how we can help you with your application. You may also engage the Ping Identity Global Client Services team for assistance with developing customizations.

Get started with the SDK

This section describes the directories and build components that comprise the SDK and provides instructions for setting up a development environment.

SDK directory structure

The PingAccess SDK directory (`PA_HOME/sdk`) contains these files and directories.

- `README.md` – Contains an overview of the SDK contents.
- `/samples/README.md` – Contains an overview of the steps necessary to build and use the samples.
- `/samples/Rules` – Contains a maven project with example plug-in implementations for Rules showing a wide range of functionality. You may use these examples for developing your own implementations.
- `/samples/Rules/README.md` – Contains the details of the Rules samples.
- `/samples/SiteAuthenticator` – Contains a maven project with example plug-in implementations for Site Authenticators. You may use these examples for developing your own implementations.
- `/samples/SiteAuthenticator/README.md` – Contains the details of the Site Authenticator samples.
- `/samples/IdentityMappings` – Contains a maven project with example plug-in implementations for Identity Mappings. You may use these examples for developing your own implementations.
- `/samples/IdentityMappings/README.md` – Contains the details of the IdentityMappings samples.
- `/samples/LoadBalancingStrategies` – Contains a maven project with example plug-in implementations for Load Balancing Strategies. You may use these examples for developing your own implementations.
- `/samples/LoadBalancingStrategies/README.md` – Contains the details of the LoadBalancingStrategies samples.
- `/samples/LocaleOverrideService` – Contains a maven project with example plug-in implementations for the Locale Override Service. You may use these examples for developing your own implementations.
- `/samples/LocaleOverrideService/README.md` – Contains the details of the LocaleOverrideService samples.
- `/apidocs/` – Contains the SDK Javadocs. Open `index.html` to get started.

SDK prerequisites

These prerequisites must be met before you use the Addon SDK for Java.

Before you start, ensure you have the Java SDK and *Apache Maven* installed. The samples use Apache Maven and assume that the PingAccess SDK can be referenced as a dependency. They reference Ping Identity's public maven repository, located at:

```
http://maven.pingidentity.com/release
```

If Internet access is unavailable, update the `pingaccess-sdk` dependency in your `pom.xml` to point to the local installation.

```
<dependency>
        <groupId>com.pingidentity.pingaccess</groupId>
        <artifactId>pingaccess-sdk</artifactId>
        <version>4.0.1.3</version>
        <scope>system</scope>
        <systemPath><PA_HOME>/lib/pingaccess-sdk-4.2.0.0.jar</systemPath>
</dependency>
<dependency>
        <groupId>javax.validation</groupId>
        <artifactId>validation-api</artifactId>
        <version>1.0.0.GA</version>
  <scope>system</scope>
        <systemPath>PA_HOME/lib/validation-api-1.0.0.GA.jar</systemPath>
</dependency>
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>1.7.4</version>
  <scope>system</scope>
        <systemPath>PA_HOME/lib/slf4j-api-1.7.4.jar</systemPath>
</dependency>
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.7.4</version>
  <scope>system</scope>
        <systemPath>PA_HOME/lib/slf4j-log4j12-1.7.4.jar</systemPath>
</dependency>
```

Replace *PA_HOME* with the path to the PingAccess installation.

How to install the SDK samples
You can install rule and site authenticator SDK samples.

- Before you begin, ensure you have the Java SDK and Apache Maven installed.
- Each sample type is installed separately:
  - For the Rules samples, navigate to `PA_HOME/sdk/samples/Rules`
  - For the Site Authenticators samples, navigate to `PA_HOME/sdk/samples/SiteAuthenticator`
- From the sample's directory, run the command: `$ mvn install`
  - This builds the samples, runs their tests, and copies the resulting jar file from the target directory to the `PA_HOME/lib` directory.

```
jsmith-MBP-2:Rules jsmith$ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder
 org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuilder
 with a thread count of 1
[INFO]
[INFO]
 ------------------------------------------------------------------------
```

```
[INFO] Building PingAccess :: Sample Rules 3.0.0-RC5
[INFO]
 ------------------------------------------------------------------------
Downloading: http://...
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @
 sample-rules ---
[INFO] Using 'ISO-8859-1' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ sample-
rules ---
[INFO] Compiling 7 source files to /Users/jsmith/Downloads/pingaccess-3.0.0-
RC5/sdk/samples/Rules/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources)
 @ sample-rules ---
[INFO] Using 'ISO-8859-1' encoding to copy filtered resources.
[INFO] Copying 4 resources
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @
 sample-rules ---
[INFO] Compiling 4 source files to /Users/jsmith/Downloads/pingaccess-3.0.0-
RC5/sdk/samples/Rules/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ sample-rules
 ---
[INFO] Surefire report directory: /Users/jsmith/Downloads/pingaccess-3.0.0-
RC5/sdk/samples/Rules/target/surefire-reports
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.pingidentity.pa.sample.TestAllUITypesAnnotationRule
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.912 sec
Running com.pingidentity.pa.sample.TestIllustrateManyUITypesRule
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.029 sec
Running com.pingidentity.pa.sample.TestValidateRulesAreAvailable
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 sec
Results :
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ sample-rules ---
[INFO] Building jar: /Users/jsmith/Downloads/pingaccess-3.0.0-RC5/sdk/
samples/Rules/target/sample-rules-3.0.0-RC5.jar
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ sample-rules
 ---
[INFO] Installing /Users/jsmith/Downloads/pingaccess-3.0.0-RC5/sdk/samples/
Rules/target/sample-rules-3.0.0-RC5.jar to /Users/jsmith/.m2/repository/com/
pingidentity/pingaccess/sample-rules/3.0.0-RC5/sample-rules-3.0.0-RC5.jar
[INFO] Installing /Users/jsmith/Downloads/pingaccess-3.0.0-RC5/sdk/samples/
Rules/pom.xml to /Users/jsmith/.m2/repository/com/pingidentity/pingaccess/
sample-rules/3.0.0-RC5/sample-rules-3.0.0-RC5.pom
[INFO]
[INFO] --- maven-antrun-plugin:1.7:run (default) @ sample-rules ---
[INFO] Executing tasks
main:
     [copy] Copying 1 file to /Users/jsmith/Downloads/pingaccess-3.0.0-RC5/
lib
[INFO] Executed tasks
[INFO]
 ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

```
[INFO]
 ------------------------------------------------------------------------
[INFO] Total time: 6.418 s
[INFO] Finished at: 2014-07-08T16:38:30-07:00
[INFO] Final Memory: 16M/38M
[INFO]
 ------------------------------------------------------------------------
```

Create your own plugins

You can create your own plugins from scratch using the Addon SDK.

Generally, the following steps are taken to implement a plugin:

1. Create a new, empty Maven project. The root directory of the Maven project is referred to as `<PLUGIN_HOME>`.
2. Copy the `pom.xml` from the appropriate sample provided in `<PA_HOME>/sdk/samples`.

   Note: For example, if you were creating a Rule, you would copy the `pom.xml` from `<PA_HOME>/sdk/samples/Rules/` to `<PLUGIN_HOME>/`.
3. Modify the `groupId`, `artifactId`, `name`, and `version` in the copied `pom.xml` file as appropriate.
4. Create a Java class that implements the plugin interface from the SDK in the `<PLUGIN_HOME>/src/main/java/com/yourpackagename` directory. This interface is referred to as a Service Provider Interface (SPI).

   Note: For example, if you are implementing a custom Rule, the class should implement the RuleInterceptor SPI.

   For each SPI, base classes are provided that simplify the implementation of the SPI.
5. Create a provider-configuration file for the plugin SPI containing the fully-qualified class name for the class created in the previous step.

   Note: For example, if you are implementing a custom Rule, you'll create a file called `<PLUGIN_HOME>/META-INF/services/com.pingidentity.pa.sdk.policy.RuleInterceptor` and its contents will be the FQCN of the class.
6. Build the Maven project to obtain a jar containing the plugin implementation.
7. Copy the jar to `<PA_HOME>/lib`.

   Important: After copying a custom plugin jar to the PingAccess lib, a restart of PingAccess is required to complete the deployment of the custom plugin.

The following sections provide the details required to complete these steps for each type of plugin:

- *Rule details*
- *Site authenticator details*
- *Identity mapping details*
- *Load balancing strategy details*
- *Locale override service details*

Rule details

If you do not need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

```
com.pingidentity.pa.sdk.policy.RuleInterceptor
```

Provider-configuration file

`<PLUGIN_HOME>/META-INF/services/ com.pingidentity.pa.sdk.policy.RuleInterceptor`

Base Classes

`com.pingidentity.pa.sdk.policy.RuleInterceptorBase`

If you need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.policy.AsyncRuleInterceptor`

Provider-configuration file

`<PLUGIN_HOME>/META-INF/services/ com.pingidentity.pa.sdk.policy.AsyncRuleInterceptor`

Base Classes

`com.pingidentity.pa.sdk.policy.AsyncRuleInterceptorBase`

Site authenticator details

If you do not need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.siteauthenticator.SiteAuthenticatorInterceptor`

Provider-configuration file

`<PLUGIN_HOME>/META-INF/services/ com.pingidentity.pa.sdk.siteauthenticator.SiteAuthenticatorInterceptor`

Base Classes

`com.pingidentity.pa.sdk.siteauthenticator.SiteAuthenticatorInterceptorBase`

If you need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.siteauthenticator.AsyncSiteAuthenticatorInterceptor`

Provider-configuration file

`<PLUGIN_HOME>/META-INF/services/ com.pingidentity.pa.sdk.siteauthenticator.AsyncSiteAuthenticatorInterceptor`

Base Classes

`com.pingidentity.pa.sdk.siteauthenticator.AsyncSiteAuthenticatorInterceptorBase`

Identity mapping details

If you do not need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.identitymapping.IdentityMappingPlugin`

Provider-configuration file

`<PLUGIN_HOME>/META-INF/services/ com.pingidentity.pa.sdk.identitymapping.IdentityMappingPlugin`

Base Classes

`com.pingidentity.pa.sdk.identitymapping.IdentityMappingPluginBase`

`com.pingidentity.pa.sdk.identitymapping.header.HeaderIdentityMappingPlugin`

If you need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.identitymapping.AsyncIdentityMappingPlugin`

Provider-configuration file

```
<PLUGIN_HOME>/META-INF/services/
com.pingidentity.pa.sdk.identitymapping.AsyncIdentityMappingPlugin
```

Base Classes

`com.pingidentity.pa.sdk.identitymapping.AsyncIdentityMappingPluginBase`

Load balancing strategy details

If you do not need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.ha.lb.LoadBalancingPlugin`

Provider-configuration file

```
<PLUGIN_HOME>/META-INF/services/
com.pingidentity.pa.sdk.ha.lb.LoadBalancingPlugin
```

Base Classes

`com.pingidentity.pa.sdk.ha.lb.LoadBalancingPluginBase`

If you need to integrate with a Third-Party Service, use the following SPIs and base classes:

SPI

`com.pingidentity.pa.sdk.ha.lb.AsyncLoadBalancingPlugin`

Provider-configuration file

```
<PLUGIN_HOME>/META-INF/services/
com.pingidentity.pa.sdk.ha.lb.AsyncLoadBalancingPlugin
```

Base Classes

`com.pingidentity.pa.sdk.ha.lb.AsyncLoadBalancingPluginBase`

Locale override service details

A Locale Override Service cannot integrate with a Third-Party Service so the following SPIs and base classes are used for all implementations:

SPI

`com.pingidentity.pa.sdk.localization.LocaleOverrideService`

Provider-configuration file

```
<PLUGIN_HOME>/META-INF/services/
com.pingidentity.pa.sdk.localization.LocaleOverrideService
```

Base Classes

No base classes are provided

Integrating with third-party services

The Addon SDK includes the ability for a custom plugin to integrate with external, third-party services via HTTP.

This section provides a high-level overview of utilizing this functionality from a custom plugin.

- *Obtaining the HTTP client instance* on page 343
- *Obtaining a handle to a third-party service* on page 343
- *Making a HTTP call to a third-party service* on page 344

- *Base classes* on page 345
- *Sample plugins* on page 345

Obtaining the HTTP client instance

PingAccess provides access to a HTTP client utility interface, HttpClient, via dependency injection. Plugins are expected to obtain an instance of this interface using an approach like the following:

```
public class DocumentationPlugin // interfaces and base classes omitted for
 brevity
{
    private HttpClient httpClient;

    // ... other code omitted ...

    @Inject
    public void setHttpClient(HttpClient httpClient)
    {
        this.httpClient = httpClient;
    }

    // ... other code omitted ...
}
```

Obtaining a handle to a third-party service

Given a HttpClient instance, a plugin will also need a handle to a Third-Party Service to make an outbound HTTP call to the service represented by the Third-Party Service administrative configuration object. This handle is an instance of the ThirdPartyServiceModel class and is specified to the HttpClient in its send method.

There are two different ways to obtain a ThirdPartyServiceModel instance:

- *Administrator-configured third-party services*
- *Third-party services for the OAuth authorization server and OIDC provider*

Administrator-configured third-party services

The PingAccess Administrative UI and API allow administrators to define the communication configuration for an external service by defining a Third-Party Service. These configuration objects can then be associated with custom plugins via their configuration.

To enable a plugin's configuration to reference a Third-Party Service, it should define a field in the configuration with the type of ThirdPartyServiceModel. Here is an example:

```
    private static class Configuration extends SimplePluginConfiguration
    {
        // ... other code omitted ...

        @UIElement(order = 30,
                type = ConfigurationType.SELECT,
                label = "Risk Authorization Service",
                modelAccessor = ThirdPartyServiceModelAccessor.class,
                required = true)
        @NotNull
        private ThirdPartyServiceModel riskAuthzService;

        // ... other code omitted ...

        public ThirdPartyServiceModel getRiskAuthzService()
        {
            return riskAuthzService;
        }

        public void setRiskAuthzService(ThirdPartyServiceModel
 riskAuthzService)
        {
            this.riskAuthzService = riskAuthzService;
        }
    }
```

The important items in this example:

- The modelAccessor attribute of the UIElement must be set to ThirdPartyServiceModelAccessor
- The field in the plugin configuration class must be of type ThirdPartyServiceModel

Third-party services for the OAuth authorization server and OIDC provider

In addition to providing a way for an administrator to configure a plugin to use an arbitrary Third-Party Service, PingAccess allows a plugin to use a third-party service that represents the OAuth Authorization Server or OIDC Provider. The benefit of leveraging this functionality is that a plugin can require access to either of these services without requiring the administrator to configure the plugin to use those services.

Similar to the previous section, the plugin obtains a ThirdPartyServiceModel instance that is a handle to the OAuth Authorization Server or OIDC Provider by indicating this requirement in its plugin configuration class. However, the mechanism is a bit different, as shown in the following example:

```
    private static class Configuration extends SimplePluginConfiguration
    {
        // ... other code omitted ...

        private ThirdPartyServiceModel oidcProvider;

        // ... other code omitted ...


        public ThirdPartyServiceModel getOidcProvider()
        {
            return oidcProvider;
        }

        @Inject
        @OidcProvider
        public void setOidcProvider(ThirdPartyServiceModel oidcProvider)
        {
            this.oidcProvider = oidcProvider;
        }
    }
```

The setter for the oidcProvider field is annotated with the `@OidcProvider` annotation.

If the `@OidcProvider` annotation includes a parameter, that parameter specifies a required endpoint defined by the OIDC Provider metadata of the current Token Provider. When the plugin is instantiated, the validation for the `@OidcProvider` parameter will pass only if the specified endpoint is a valid HTTP URI in the OIDC Provider Metadata. For example, this annotation will require the `backchannel_authentication` URI:

```
@OidcProvider(“backchannel_authentication”)
```

Making a HTTP call to a third-party service

With an instance of HttpClient and an instance of ThirdPartyServiceModel in hand, a plugin can make a HTTP call to the external service represented by the ThirdPartyServiceModel. Here is an example method that makes a GET request to a resource on the external service with a path of `/data` and a query string of `page=1`:

```
private static CompletionStage<ClientResponse> sendRequest(HttpClient
 httpClient,

 ThirdPartyServiceModel model)
{
    Headers headers = ClientRequest.createHeaders();
    headers.setAccept(Collections.singletonList("application/json"));

    ClientRequest request = new ClientRequest(Method.GET,
                                              "/data?page=1",
                                              headers);

    return httpClient.send(request, model);
}
```

The result of the `HttpClient send` method is a CompletionStage. A CompletionStage is returned because PingAccess is performing the HTTP call asynchronously and as a result, handling of the result of the call needs to be performed by callbacks registered with the CompletionStage.

You can use the `getRequestUri()` method to stand in for the endpoint. This can be useful if the endpoint is not known during development. For example:

```
    ClientRequest request = new ClientRequest(Method.GET,
                                              model.getRequestUri().get(),
                                              headers);
```

The `RequestUri` is set by the `@OidcProvider("RequestUri")` annotation, and is unset if the `@OidcProvider("RequestUri")` annotation is not present.

For a more complete example of using the HttpClient to make an external HTTP call, refer to the sample SDK plugins packaged with the PingAccess distribution.

Base classes

The SDK provides the following base classes to make it easier to implement a plugin that leverages the HttpClient interface. They all provide access to a HttpClient instance via a getHttpClient method.

- `AsyncRuleInterceptorBase`
- `AsyncSiteAuthenticatorInterceptorBase`
- `AsyncIdentityMappingPluginBase`
- `AsyncLoadBalancingPluginBase`

Sample plugins

The use of the HttpClient and ThirdPartyServiceModel classes are demonstrated in the following samples provided in the PingAccess distribution:

- RiskAuthorizationRule, a Rule that obtains a risk score from an external, risk service as well as leveraging the OAuth Authorization Server to obtain an OAuth access token used to access the risk service.
- MetricBasedPlugin, a Load Balancing Strategy that obtains host capacity metadata from an external service.

Implementation guidelines

These 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 as well as additional functionality.

ⓘ **Important:** A restart of PingAccess is required after the deployment of any custom plugins written in Java.

Logging

Use the SLF4j API for logging activities in your module. Documentation on using SLF4j is available on the *SLF4j website*.

Lifecycle

The plugins and the implementation of a PluginConfiguration can be instantiated for a number of reasons and at many times. For example, with a RuleInterceptor here is what happens before the RuleInterceptor is available to process user requests:

- The Rule annotation on the implementation class of the RuleInterceptor is interrogated to determine which PluginConfiguration instance will be instantiated.
- The following is performed on RuleInterceptor and PluginConfiguration. Which of these is handled first is not defined.
  - The bean will be provided to Spring for Autowiring.
  - The bean will be provided to Spring for post construction initialization. (See PostConstruct)
- PluginConfiguration.setName(String) is called.
- PA attempts to map the incoming JSON configuration to the PluginConfiguration instance.
- ConfigurablePlugin.configure(PluginConfiguration) is called.
- Validator.validate(Object, Class[]) method is invoked and provided to the RuleInterceptor.
- The instance is then made available to service end user requests, such as RequestInterceptor.handleRequest(com.pingidentity.pa.sdk.http.Exchange) and ResponseInterceptor.handleResponse(com.pingidentity.pa.sdk.http.Exchange)

Injection

Before they are put into use, Rules, SiteAuthenticators, and their defined PluginConfigurations are passed through Spring's Autowiring and initialization. To future-proof any code against changes in PingAccess, we recommend that Spring not be used as a dependency. Use the annotation javax.inject.Inject for any injection.

Classes available for injection

Currently, injection is available for the following classes:

- com.pingidentity.pa.sdk.util.TemplateRenderer

Differences between Rules for Agents and Sites

Rules may be applied to applications associated with Agents or Sites. Some features of the SDK are not available to rules that are applied to agents. Rules that use features only available to sites should be marked as only applying to sites. This is done by setting the destination element of the rule annotation to the value {RuleInterceptorSupportedDestination.Site}

Rules that apply only to agents are limited in the following ways:

- The handleResponse method is not called.
- The request body is not present.
- The Exchange.getDestinations list is empty and modifying the destination list has no effect.

As with rules that use features only available to sites, rules that only apply to agents should be marked as only applying to agents. To do this, set the destination element of the rule annotation to the value {RuleInterceptorSupportedDestination.Agent}.

PingAccess Addon SDK for Java Migration Guide

Plugins built against the Java Addon SDK for PingAccess 5.0 are now required to be built with JDK 8, as the SDK for PingAccess 5.0 uses Java 8 features. Previously, plugins could be built with either JDK 7 or JDK 8.

The following sections provide a detailed description of the changes, organized by package. Where relevant, code examples show how to port existing code to account for the changes in the SDK APIs.

General changes

Prevent modification to Request in Response chain

Starting in PingAccess 5.0, any modifications made to a Request or its header fields during response processing will now result in a warning log message and the modification operation being ignored. Previously, PingAccess would log a warning message about the modification but still allow the modification operation to complete.

Retrieving Key Pair and Trusted Certificate Group configuration data

In the previous version of the SDK, a SDK plugin accessed the configuration data of a Key Pair or Trusted Certificate Group configured via the Administrative API by annotating a field in the plugin's PluginConfiguration class with a JsonDeserialize annotation, specifying the appropriate custom deserializer from the SDK. For example:

```
public class Configuration extends SimplePluginConfiguration
{
   @JsonDeserialize(using = PrivateKeyDeserializer.class)
    KeyStore.PrivateKeyEntry keyPair;

   @JsonDeserialize(using = TrustedCertificateGroupDeserializer.class)
    Collection<X509Certificate> certificateGroup;
}
```

In the current version of the SDK, this mechanism has changed to be less error-prone as well as to provide access to more properties of the Key Pairs and Trusted Certificate Groups. The previous Configuration class should be ported to the following:

```
public class Configuration extends SimplePluginConfiguration
{
   KeyPairModel keyPair;

   TrustedCertificateGroupModel certificateGroup;
}
```

The KeyPairModel#getPrivateKeyEntry method provides access to the KeyStore.PrivateKeyEntry object for the corresponding Key Pair in the administrative configuration. The TrustedCertificateGroupModel#getCertificates method provides access to the Collection of X509Certificate objects in the corresponding Trusted Certificate Group in the administrative configuration. Refer to the JavaDoc for each of these classes for more information.

Related to this change, the provided implementations of ConfigurationModelAccessor, PrivateKeyAccessor and TrustedCertificateGroupAccessor, have been updated to use these new classes. Both classes have also been moved to new packages. PrivateKeyAccessor has also been renamed to KeyPairAccessor.

BEFORE:

```
import com.pingidentity.pa.sdk.accessor.PrivateKeyAccessor;
import com.pingidentity.pa.sdk.accessor.TrustedCertificateGroupAccessor;

// … class definition omitted ...

private void invokePrivateKeyAccessorGet(
       PrivateKeyAccessor accessor,
       String id)
{
   KeyStore.PrivateKeyEntry keyPair = accessor.get(id);
}
private void invokeTrustedCertificateGroupAccessorGet(
       TrustedCertificateGroupAccessor accessor,
       String id)
{
   Collection<X509Certificate> certificates = accessor.get(id);
}
```

AFTER:

```
import
 com.pingidentity.pa.sdk.accessor.certgroup.TrustedCertificateGroupModel;
import com.pingidentity.pa.sdk.accessor.keypair.KeyPairAccessor;

// … class definition omitted ...

private void invokePrivateKeyAccessorGet(
       KeyPairAccessor accessor,
       String id)
{
   KeyStore.PrivateKeyEntry keyPair = accessor.get(id)

 .map(KeyPairModel::getPrivateKeyEntry)
                                              .orElse(null);
}

private void invokeTrustedCertificateGroupAccessorGet(
       TrustedCertificateGroupAccessor accessor,
       String id)
{
   Collection<X509Certificate> certificates = accessor.get(id)
                  .map(TrustedCertificateGroupModel::getCertificates)
                  .orElse(null);
}
```

Changes to validation of PluginConfiguration instances

In the previous version of the SDK, the ConfigurablePlugin#configure method was invoked and passed a PluginConfiguration instance. The ConfigurablePlugin was expected to assign the specified PluginConfiguration instance to a field annotated with the javax.validation.Valid annotation. After the configure method returned, PingAccess passed the ConfigurablePlugin instance to a javax.validation.Validator for further validation.

If setup correctly, this logic allows javax.validation.Constraint annotations to be used to declare the validation to be applied to fields in a PluginConfiguration implementation, ensuring the configuration was valid as well as providing validation error message to PingAccess to provide to administrators using the Administrative API or UI.

However, if the ConfigurablePlugin#configure method needed to post-process the specified PluginConfiguration instance, the method needed to duplicate all the validation declared on the fields of the PluginConfiguration.

To remove the need for this duplication of validation logic, PingAccess will now validate the PluginConfiguration instance with a javax.validation.Validator prior to passing the instance to the ConfigurablePlugin#configure method.

Further, the ConfigurablePlugin no longer needs to annotate the field used to hold the PluginConfiguration instance. The field is still necessary to implement the ConfigurablePlugin#getConfiguration method.

The following example ConfigurablePlugin implementation demonstrates this change:

```
public class ValidationExample
       implements ConfigurablePlugin<ValidationExample.Configuration>
{
   // @Valid annotation no longer required
   private Configuration configuration;

   @Override
   public void configure(Configuration configuration) throws
 ValidationException
   {
       this.configuration = configuration;

       // With the previous version of the SDK, these assertions were not
       // guaranteed to be true, despite the javax.validation.Constraint
       // annotations enforcing these conditions.
       //
       // In the current version of the SDK, these assertions are guaranteed
       // to be true because they are enforced by the
 javax.validation.Constraint
       // annotations on the fields in the PluginConfiguration class, and
 the
       // PluginConfiguration validation is performed before invoking the
       // configure method.
       //
       // The end result is that plugins can remove duplicated validation
       // logic from the configure method if further post-processing of the
       // configuration needs to be performed.
       assert(configuration.getAttributeName() != null);
       assert(configuration.getAttributeName().length() > 0);
       assert(configuration.getAttributeName().length() <= 16);
       assert(configuration.getAttributeValue() != null);

   }

   @Override
   public Configuration getConfiguration()
   {
       return configuration;
   }

   static class Configuration extends SimplePluginConfiguration
   {
       @NotNull
       @Size(min = 1,
             max = 16,
             message = "Attribute name length must be between 1 and 16
 characters")
       private String attributeName;

       @NotNull
       private String attributeValue;

       public String getAttributeName()
       {
           return attributeName;
       }

       public void setAttributeName(String attributeName)
       {
           this.attributeName = attributeName;
       }

       public String getAttributeValue()
       {
           return attributeValue;
       }

       public void setAttributeValue(String attributeValue)
       {
           this.attributeValue = attributeValue;
       }
   }
}
```

com.pingidentity.pa.sdk.http

com.pingidentity.pa.sdk.http.Body

The Body interface has changed to require an explicit read of data before invoking methods to obtain that data. Previously, methods to obtain the data would result in an implicit read of the data. The following code examples illustrate this change in semantics.

As the updated JavaDoc for the Body interface indicates, plugins should avoid interrogating a Body object unless absolutely necessary because reading a Body object's data into memory can impact the scalability of PingAccess. As plugin code is updated, evaluate whether the Body object needs to be used by the plugin.

Using the Body#read method

BEFORE:

```
private void invokeRead(Body body) throws IOException
{
    body.read();
}
```

AFTER:

```
private void invokeRead(Body body) throws AccessException
{
    try
    {
        body.read();
    }
    catch (IOException e)
    {
        throw new AccessException("Failed to read body content",
                                  HttpStatus.BAD_GATEWAY,
                                  e);
    }
}
```

Using the Body#getContent method:

BEFORE:

```
private void invokeGetContent(Body body) throws IOException
```

```
{
    byte[] content = body.getContent();
}
```

AFTER:

```
private void invokeGetContent(Body body) throws AccessException
{
    invokeRead(body); // see the Body#read code example for this
 method
    byte[] content = body.getContent();
}
```

Using the Body#getBodyAsStream method:

BEFORE:

```
private void invokeGetBodyAsStream(Body body) throws IOException
{
    InputStream stream = body.getBodyAsStream();
}
```

AFTER:

```
private void invokeGetBodyAsStream(Body body) throws
 AccessException
{
    invokeRead(body); // see the Body#read code example for this
 method
    InputStream stream = body.newInputStream();
}
```

Note the rename of the method from `getBodyAsStream` to `newInputStream`.

Using the Body#write method:

BEFORE:

```
private void invokeWrite(Body body, BodyTransferrer
 bodyTransferrer) throws IOException

{

    body.write(bodyTransferrer);

}
```

AFTER:

This functionality is no longer supported. To obtain the content of the Body, read the content into memory using the Body#read method and then invoke Body#getContent or Body#newInputStream.

Using the Body#getLength method

BEFORE:

```
private void invokeGetLength(Body body) throws IOException
{
    int length = body.getLength();
}
```

AFTER:

```
private void invokeGetLength(Body body) throws AccessException

{

    invokeRead(body); // see the Body#read code example for this
 method

    int length = body.getLength();

}
```

Using the Body#getRaw method:

BEFORE:

```
private void invokeGetRaw(Body body) throws IOException
{
    byte[] rawBody = body.getRaw();
}
```

AFTER:

This functionality is no longer supported. This method used to provide access to the content as it appeared on the wire, which required complicated handling if the body content used a chunked Transfer-Encoding. Use Body#getContent instead.

com.pingidentity.pa.sdk.http.BodyFactory

Using the BodyFactory#continuousBody method

BEFORE:

```
private void invokeContinuousBody(BodyFactory bodyFactory, byte[]
 content)
{
    Body body = bodyFactory.continuousBody(content);
}
```

AFTER:

```
private void invokeContinuousBody(BodyFactory bodyFactory, byte[]
 content)
{
    Body body = bodyFactory.createInMemoryBody(content);
}
```

BEFORE:

```
private void invokeContinuousBody(BodyFactory bodyFactory,
 InputStream in)
{
    Body body = bodyFactory.continuousBody(in);
}
```

AFTER:

A Body instance can no longer be created from an InputStream using the BodyFactory class. Instead, a plugin should read the contents of the InputStream into a byte array and provide the byte array to BodyFactory#createInMemoryBody.

com.pingidentity.pa.sdk.http.Constants

The constants available from this class have been removed from the SDK. Plugins using these constants should maintain their own constants with the needed values.

com.pingidentity.pa.sdk.http.Exchange

A handful of methods have been removed from the Exchange.

Further, the mechanism for storing data on the Exchange via properties has been enhanced to make it easier to write type-safe code when working with Exchange properties.

Using the Exchange#getCreationTime method

BEFORE:

```
Calendar creationTime = exchange.getCreationTime();
```

AFTER:

```
Calendar creationTime = Calendar.getInstance();
creationTime.setTime(Date.from(exchange.getCreationTime()));
```

NOTE: If a Calendar object is not required, consider using the Instant object returned from the getCreationTime method directly instead of converting it into a Calendar object.

Using the Exchange#getDestinations method

BEFORE:

```
List<String> destinations = exchange.getDestinations();
```

AFTER:

This functionality is no longer supported. Consider using the Exchange#getTargetHosts method to obtain similar information from the Exchange.

Using the Exchange#getOriginalHostHeader method

BEFORE:

```
String originalHostHeader = exchange.getOriginalHostHeader();
```

AFTER:

This functionality is no longer supported. Consider using the Exchange#getUserAgentHost method to obtain similar information from the Exchange. The getUserAgentHost method leverages the PingAccess HTTP requests configuration to determine the Host header value sent by the user agent.

Using the Exchange#getOriginalHostHeaderHost method

BEFORE:

```
String host = exchange.getOriginalHostHeaderHost();
```

AFTER:

This functionality is no longer supported. Consider using the Exchange#getUserAgentHost method to obtain similar information from the Exchange. The getUserAgentHost method leverages the PingAccess HTTP requests configuration to determine the Host header value sent by the user agent.

Using the Exchange#getOriginalHostHeaderPort method

BEFORE:

```
String port = exchange.getOriginalHostHeaderPort();
```

AFTER:

This functionality is no longer supported. Consider using the Exchange#getUserAgentHost method to obtain similar information from the Exchange. The getUserAgentHost method leverages the PingAccess HTTP requests configuration to determine the Host header value sent by the user agent.

Using the Exchange#getOriginalRequestBaseUri method

BEFORE:

```
String originalRequestBaseUri =
 exchange.getOriginalRequestBaseUri();
```

AFTER:

This functionality is no longer supported. A possible replacement is as follows:

```
String originalRequestBaseUri = exchange.getUserAgentProtocol() +
                                "://" +
                                exchange.getUserAgentHost();
```

Using the Exchange#getProperties method

BEFORE:

```
Map<String, String> properties = exchange.getProperties();
```

AFTER:

This functionality is no longer supported. Properties should be obtained individually from the Exchange.

Using the Exchange#getRequestBaseUri method

BEFORE:

```
String requestBaseUri = exchange.getRequestBaseUri();
```

AFTER:

This functionality is no longer supported. A possible replacement is as follows:

```
String requestBaseUri = exchange.getUserAgentProtocol() +
                        "://" +
                        exchange.getUserAgentHost();
```

Using the Exchange#getRequestScheme method

BEFORE:

```
String requestScheme = exchange.getRequestScheme();
```

AFTER:

This functionality is no longer supported. A possible replacement is as follows:

```
String requestScheme = exchange.getUserAgentProtocol() + "://";
```

Using the Exchange#getUser method

BEFORE:

```
User user = exchange.getUser();
```

AFTER:

The User interface is no longer supported. Use the Identity interface instead. It can be retrieved via the Exchange#getIdentity method.

Using the Exchange#setUser method

BEFORE:

```
private void invokeSetUser(Exchange exchange, User user)
{
    exchange.setUser(user);
}
```

AFTER:

This functionality is no longer supported. The identity associated with an Exchange cannot be replaced.

Using the Exchange#setSourceIp method

BEFORE:

```
private void invokeSetSourceIp(Exchange exchange, String
 sourceIp)
{
    exchange.setSourceIp(sourceIp);
}
```

AFTER:

This functionality is no longer supported. This value cannot be changed.

Using the Exchange#setProperty method

BEFORE:

```
private void invokeSetProperty(Exchange exchange, String
 propertyKey, String value)
{
    exchange.setProperty(propertyKey, value);
}
```

AFTER:

```
private void invokeSetProperty(Exchange exchange,
                               ExchangeProperty<String>
 propertyKey,
                               String value)
{
    exchange.setProperty(propertyKey, value);
}
```

See the JavaDoc for ExchangeProperty for instructions on creating an ExchangeProperty object.

Using the Exchange#getProperty method

BEFORE:

```
private void invokeGetProperty(Exchange exchange, String
 propertyKey)
{
    Object propertyValueObj = exchange.getProperty(propertyKey);
    if (propertyValueObj instanceof String)
    {
        String propertyValue = (String) propertyValueObj;
    }
}
```

AFTER:

```
private void invokeGetProperty(Exchange exchange,
 ExchangeProperty<String> propertyKey)
{
    String propertyValue =
 exchange.getProperty(propertyKey).orElse(null);
}
```

NOTE: Exchange#getProperty now returns an Optional object instead of the Object directly.

com.pingidentity.pa.sdk.http.Header

This deprecated class has been replaced by the Headers interface. A Headers object can be created via a HeadersFactory obtained from the ServiceFactory#headersFactory method. The majority of methods on Header have counterparts on the Headers interface. See the JavaDoc for the Headers interface for more information.

com.pingidentity.pa.sdk.http.HeaderField

This class is now final and cannot be extended.

Constructing a HeaderField

BEFORE:

```
private HeaderField createHeaderField(String line)
{
    return new HeaderField(line);
}
```

AFTER:

```
private HeaderField createHeaderField(String line)
{
    String name = line.substring(0, line.indexOf(':'));
    String value = (line.substring(line.indexOf(":") +
 1)).trim();

    return new HeaderField(name, value);
}
```

NOTE: Parsing an HTTP header field line can be error prone, consider if the plugin can be avoid having to parse an HTTP header field line.

Using the HeaderField#setHeaderName method:

BEFORE:

```
private void invokeSetHeaderName(HeaderField field)
```

```
{
    field.setHeaderName(new HeaderName("X-Custom"));
}
```

AFTER:

This functionality is no longer supported. A HeaderField's name is set upon construction and cannot be changed.

Using the HeaderField#getApproximateSize method:

BEFORE:

```
int approximateSize = field.getApproximateSize();
```

AFTER:

This method has been removed. The value returned by the method can still be computed:

```
int approximateSize = 2 * (4 +

 field.getHeaderName().toString().length() +
                          field.getValue().length());
```

com.pingidentity.pa.sdk.http.Headers

A few methods on the Headers interface have been updated to use the Instant class, instead of Date.

Using the Headers#getDate method

BEFORE:

```
Date date = headers.getDate();
```

AFTER:

```
Date date = Date.from(headers.getDate());
```

Using the Headers#setDate method

BEFORE:

```
private void invokeSetDate(Headers headers, Date date)
{
    headers.setDate(date);
}
```

AFTER:

```
private void invokeSetDate(Headers headers, Date date)
{
    headers.setDate(date.toInstant());
}
```

Using the Headers#getLastModified method

BEFORE:

```
SimpleDateFormat format = new SimpleDateFormat("E, dd MMM yyyy
 HH:mm:ss z",
                                                Locale.ENGLISH);
```

```
String lastModified = headers.getLastModified();
if (lastModified != null)
{
    Date lastModifiedDate = format.parse(lastModified);
}
```

AFTER:

```
Date lastModifiedDate = Date.from(headers.getLastModified());
```

Using the Headers#setLastModified method

BEFORE:

```
private void invokeSetLastModified(Headers headers, Date date)
{
    SimpleDateFormat format = new SimpleDateFormat("E, dd MMM
 yyyy HH:mm:ss z",

 Locale.ENGLISH);

    headers.setLastModified(format.format(date));
}
```

AFTER:

```
private void invokeSetLastModified(Headers headers, Date date)
{
    headers.setLastModified(date.toInstant());
}
```

com.pingidentity.pa.sdk.http.HeadersFactory

Using the HeadersFactory#createFromRawHeaderFields method

BEFORE:

```
private void invokeCreateFromRawHeaderFields(HeadersFactory
 factory,
                                             List<String> fields)
 throws ParseException
{
    Headers headers = factory.createFromRawHeaderFields(fields);
}
```

AFTER:

This functionality is no longer supported. Consider if the plugin can create HeaderFields directly and utilize the HeadersFactory#create method.

com.pingidentity.pa.sdk.http.HttpStatus

The HttpStatus enum was converted to a final class. Common HttpStatus instances are defined as constants on HttpStatus.

Using the HttpStatus#getLocalizationKey method

BEFORE:

```
String localizationKey = status.getLocalizationKey();
```

AFTER:

This functionality is no longer supported. Instead, a HttpStatus contains a LocalizedMessage instance that encapsulates the localization of the status message for use in error templates.

com.pingidentity.pa.sdk.http.MimeType

The constants available in this class are now available as constant MediaType instances in the class com.pingidentity.pa.sdk.http.CommonMediaTypes.

com.pingidentity.pa.sdk.http.MediaType

This class is now final and cannot be extended.

Constructing a MediaType

BEFORE:

```
private void createMediaType(String mediaTypeString)
{
    MediaType mediaType = new MediaType(mediaTypeString);
}
```

AFTER:

```
private void createMediaType(String mediaTypeString)
{
    MediaType mediaType = MediaType.parse(mediaTypeString);
}
```

com.pingidentity.pa.sdk.http.Message

A number of methods have been removed from the Message interface.

Using the Message#getBodyAsStream method

BEFORE:

```
InputStream bodyStream = message.getBodyAsStream();
```

AFTER:

This functionality is no longer supported. However, the following code snippet could be used to maintain semantics of the old method.

```
Body body = message.getBody();
try
{
    body.read();
}
catch (IOException | AccessException e)
{
    throw new RuntimeException("Could not get body as stream",
 e);
}

InputStream bodyStream = body.newInputStream();
```

While this snippet maintains semantics, it is recommended that a plugin propagate errors as an AccessException instead of a RuntimeException.

Using the Message#getCharset method

BEFORE:

```
Charset charset = message.getCharset();
```

AFTER:

This functionality is no longer supported. However, the following code snippet could be used to maintain semantics of the old method.

```
Charset charset = message.getHeaders().getCharset();
if (charset == null)
{
    charset = StandardCharsets.UTF_8;
}
```

While this snippet maintains semantics, a plugin should consider how to handle the case where a Charset is not specified by a Message's header fields. Assuming a Charset of UTF-8 could lead to issues in some cases.

Using the Message#getHeader method

BEFORE:

```
Header header = message.getHeader();
```

AFTER:

This functionality is no longer supported. Instead, use Message#getHeaders and the Headers interface instead of Header.

Using the Message#setHeader method

BEFORE:

```
private void invokeSetHeader(Message message, Header header)
{
    message.setHeader(header);
}
```

AFTER:

This functionality is no longer supported. Instead, use Message#setHeaders and the Headers interface instead of Header.

Using the Message#isDeflate method

BEFORE:

```
boolean deflate = message.isDeflate();
```

AFTER:

This method has been removed. However, the value can still be computed with the following code snippet:

```
List<String> contentEncodingValues =
 message.getHeaders().getContentEncoding();
boolean deflate = contentEncodingValues.stream().anyMatch(v ->
 v.equalsIgnoreCase("deflate"))
                  && contentEncodingValues.size() == 1;
```

Using the Message#isGzip method

BEFORE:

```
boolean gzip = message.isGzip();
```

AFTER:

This method has been removed. However, the value can still be computed with the following code snippet:

```
List<String> contentEncodingValues =
 message.getHeaders().getContentEncoding();
boolean gzip = contentEncodingValues.stream().anyMatch(v ->
 v.equalsIgnoreCase("gzip"))
               && contentEncodingValues.size() == 1;
```

Using the Message#isHTTP10 method

BEFORE:

```
boolean http10 = message.isHTTP10();
```

AFTER:

This method has been removed. However, the value can still be computed with the following code snippet:

```
boolean http10 = message.getVersion().equals("1.0");
```

Using the Message#isHTTP11 method

BEFORE:

```
boolean http11 = message.isHTTP11();
```

AFTER:

The method has been removed. However, the value can still be computed with the following code snippet:

```
boolean http11 = message.getVersion().equals("1.1");
```

Using the Message#read method

BEFORE:

```
private void invokeRead(Message message,
                        InputStream inputStream,
                        boolean createBody) throws IOException
{
    message.read(inputStream, createBody);
}
```

AFTER:

This functionality is no longer supported. A Request attached to an Exchange can no longer be completely replaced, but individual components can be replaced, such as the method, uri, headers and body. A Response attached to an Exchange can be replaced by using Exchange#setResponse.

Using the Message#setVersion method

BEFORE:

```
private void invokeSetVersion(Message message, String version)
{
    message.setVersion(version);
}
```

AFTER:

This functionality is no longer supported. The version of a Message cannot be changed.

Using the Message#write method

BEFORE:

```
private void invokeWrite(Message message,
                         OutputStream output) throws IOException
{
    message.write(output);
}
```

AFTER:

This functionality is no longer supported. However, the following code snippet can be used to perform the equivalent operation:

```
private void invokeWrite(Message message,
                         OutputStream output) throws IOException,
 AccessException
{
    Body body = message.getBody();
    body.read();


 output.write(message.getStartLine().getBytes(StandardCharsets.ISO_8859_1));

 output.write(message.getHeaders().toString().getBytes(StandardCharsets.ISO_8859
    output.write("\r\n".getBytes(StandardCharsets.ISO_8859_1));
    output.write(body.getContent());
    output.flush();
}
```

com.pingidentity.pa.sdk.http.Method

The Method interface has been converted to a final class. Additionally, the related Methods enum has been merged into the Method class. The Method class provides common Method instances as class-level constants.

Obtaining a common Method instance

BEFORE:

```
Method get = Methods.GET
```

AFTER:

```
Method get = Method.GET;
```

Using the Method#getMethodName method

BEFORE:

```
String methodName = method.getMethodName();
```

AFTER:

```
String methodName = method.getName();
```

com.pingidentity.pa.sdk.http.Request

A few methods have been removed from the Request interface.

Using the Request#getPostParams method

BEFORE:

```
private void invokeGetPostParams(Request request) throws
 IOException
{
    Map<String, String[]> postParams = request.getPostParams();
}
```

AFTER:

```
private void invokeGetPostParams(Request request) throws
 AccessException
{
    Body body = request.getBody();
    try
    {
        body.read();
    }
    catch (IOException e)
    {
        throw new AccessException("Failed to read body content",
                                  HttpStatus.BAD_GATEWAY,
                                  e);
    }

    Map<String, String[]> postParams = body.parseFormParams();
}
```

Using the Request#isMultipartFormPost method

BEFORE:

```
boolean multipartFormPost = request.isMultipartFormPost();
```

AFTER:

This method has been removed from the Request interface. However, the value can still be calculated using the following code snippet:

```
Headers headers = request.getHeaders();

boolean multipartFormPost =
        request.getMethod() == Method.POST
        && headers.getContentType() != null
        &&
 headers.getContentType().getBaseType().equals("multipart/form-
data")
        && headers.getContentType().getParameter("boundary") !=
 null;
```

com.pingidentity.pa.sdk.http.ResponseBuilder

A handful of methods were removed from ResponseBuilder. Additionally, a handful of methods have changed their semantics, particularly those that included an HTML message payload. See the updated JavaDoc for ResponseBuilder for more info.

Using the ResponseBuilder#badRequestText method

BEFORE:

```
Response response =
 ResponseBuilder.badRequestText(message).build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.BAD_REQUEST)

 .contentType(CommonMediaTypes.TEXT_PLAIN)
                                  .body(message)
                                  .build();
```

NOTE: This approach does not localize the response body. Using a TemplateRenderer is recommended instead.

Using the ResponseBuilder#contentLength method

BEFORE:

```
Response response =
 ResponseBuilder.newInstance().contentLength(length).build();
```

AFTER:

This functionality is no longer supported. Consider using one of the ResponseBuilder#body methods instead of explicitly setting the content length. This ensures that the body content of the Response aligns with the Content-Length header field.

Using the ResponseBuilder#continue100 method

BEFORE:

```
Response response = ResponseBuilder.continue100().build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.CONTINUE).build();
```

Using the ResponseBuilder#forbiddenText method

BEFORE:

```
Response response = ResponseBuilder.forbiddenText().build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.FORBIDDEN)

 .contentType(CommonMediaTypes.TEXT_PLAIN)

 .body(HttpStatus.FORBIDDEN.getMessage())
                                  .build();
```

NOTE: This approach does not localize the response body. Using a TemplateRenderer is recommended instead.

Using the ResponseBuilder#forbiddenWithoutBody method

BEFORE:

```
Response response =
 ResponseBuilder.forbiddenWithoutBody().build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.FORBIDDEN).build();
```

BEFORE:

```
Response response =
 ResponseBuilder.forbiddenWithoutBody(message).build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.FORBIDDEN).build();
```

NOTE: In the original method, the String message parameter was not used.

Using the ResponseBuilder#htmlMessage method

BEFORE:

```
String message = ResponseBuilder.htmlMessage(caption, text);
```

AFTER:

This functionality is no longer supported. Plugins that used this method will need to construct the HTML message without this method. Consider using the TemplateRenderer utility class in place of this method.

Using the ResponseBuilder#internalServerError method

BEFORE:

```
Response response =
 ResponseBuilder.internalServerError(message).build();
```

AFTER:

```
Response response =
 ResponseBuilder.internalServerError().body(message).build();
```

NOTE: This approach does not localize the response body. Using a TemplateRenderer is recommended instead.

Using the ResponseBuilder#internalServerErrorWithoutBody method

BEFORE:

```
Response response =
 ResponseBuilder.internalServerErrorWithoutBody().build();
```

AFTER:

```
Response response =
 ResponseBuilder.internalServerError().build();
```

Using the ResponseBuilder#newInstance method

The no-arg newInstance method has been removed. A HttpStatus is required to create an instance of ResponseBuilder and the required HttpStatus object should be passed to the newInstance method that accepts a HttpStatus.

BEFORE:

```
Response response = ResponseBuilder.newInstance().build()
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.INTERNAL_SERVER_ERROR).build();
```

Using the ResponseBuilder#noContent method

BEFORE:

```
Response response = ResponseBuilder.noContent().build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.NO_CONTENT).build();
```

Using the ResponseBuilder#notFoundWithoutBody method

BEFORE:

```
Response response =
 ResponseBuilder.notFoundWithoutBody().build();
```

AFTER:

```
Response response = ResponseBuilder.notFound().build();
```

Using the ResponseBuilder#serverUnavailable method

BEFORE:

```
Response response =
 ResponseBuilder.serverUnavailable(message).build();
```

AFTER:

```
Response response =
 ResponseBuilder.serviceUnavailable().body(message).build();
```

NOTE: This approach does not localize the response body. Using a TemplateRenderer is recommended instead.

Using the ResponseBuilder#serviceUnavailableWithoutBody method

BEFORE:

```
Response response =
 ResponseBuilder.serverUnavailableWithoutBody().build();
```

AFTER:

```
Response response = ResponseBuilder.serviceUnavailable().build();
```

Using the ResponseBuilder#status method

The status methods have been removed. Instead the status should be specified to the newInstance method as it is now required.

BEFORE:

```
Response response =
 ResponseBuilder.newInstance().status(HttpStatus.OK).build();
```

AFTER:

```
Response response =
 ResponseBuilder.newInstance(HttpStatus.OK).build();
```

Using the ResponseBuilder#unauthorizedWithoutBody method

BEFORE:

```
Response response =
 ResponseBuilder.unauthorizedWithoutBody().build();
```

AFTER:

```
Response response = ResponseBuilder.unauthorized().build();
```

com.pingidentity.pa.sdk.http.Response

A few methods were removed from the Response interface.

Using the Response#isRedirect method

BEFORE:

```
boolean redirect = response.isRedirect();
```

AFTER:

```
boolean redirect = response.getStatusCode() >= 300
                   && response.getStatusCode() < 400;
```

Using the Response#setStatusCode method

BEFORE:

```
response.setStatusCode(HttpStatus.OK.getCode());
```

AFTER:

```
response.setStatus(HttpStatus.OK);
```

Using the Response#setStatusMessage method

BEFORE:

```
response.setStatusMessage(HttpStatus.OK.getMessage());
```

AFTER:

```
response.setStatus(HttpStatus.OK);
```

com.pingidentity.pa.sdk.identity

com.pingidentity.pa.sdk.identity.Identity

The getTokenExpiration method was updated to use an Instant instead of Date.

Using the Identity#getTokenExpiration method

BEFORE:

```
Date expiration = identity.getTokenExpiration();
```

AFTER:

```
Date expiration = Date.from(identity.getTokenExpiration());
```

com.pingidentity.pa.sdk.identity.OAuthTokenMetadata

The OAuthTokenMetadata methods now use an Instant instead of a Date.

Using the OAuthTokenMetadata#getExpiresAt method:

BEFORE:

```
Date expiresAt = metadata.getExpiresAt();
```

AFTER:

```
Date expiresAt = Date.from(metadata.getExpiresAt());
```

Using the OAuthTokenMetadata#getRetrievedAt method:

BEFORE:

```
Date retrievedAt = metadata.getRetrievedAt();
```

AFTER:

```
Date retrievedAt = Date.from(metadata.getRetrievedAt());
```

com.pingidentity.pa.sdk.identitymapping.header

ClientCertificateMapping has been removed from the SDK, as it was not required to create an IdentityMappingPlugin implementation.

Plugins utilizing this class should create their own version of this class.

com.pingidentity.pa.sdk.policy

com.pingidentity.pa.sdk.policy.AccessExceptionContext

The nested Builder class has been removed from AccessExceptionContext and instead AccessExceptionContext is a builder, that can be initially created with the new AccessExceptionContext#create method.

The LocalizedMessage interface has been introduced to simplify the configuration of a localized message for use in an error template. A LocalizedMessage has three implementations provided in the SDK: FixedMessage, BasicLocalizedMessage and ParameterizedLocalizedMessage. See the following code examples for more information on using these new classes.

Constructing an AccessExceptionContext:

BEFORE:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 Throwable cause)

{
    return AccessExceptionContext.builder()
                                .cause(cause)
                                .httpStatus(httpStatus)

 .exceptionMessage(httpStatus.getMessage())

 .errorDescription(httpStatus.getLocalizationKey())
                                .errorDescriptionIsKey(true)

 .errorDescriptionSubstitutions(new String[0])
                                .build();
}
```

AFTER:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 Throwable cause)
{
    return AccessExceptionContext.create(httpStatus)
                                .cause(cause)

 .exceptionMessage(httpStatus.getMessage())

 .errorDescription(httpStatus.getLocalizedMessage());
}
```

BEFORE:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 String localizationKey,

 String[] substitutions)
{
    return AccessExceptionContext.builder()
                                .httpStatus(httpStatus)

 .errorDescription(localizationKey)
                                .errorDescriptionIsKey(true)

 .errorDescriptionSubstitutions(substitutions)
                                .build();
}
```

AFTER:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 String localizationKey,

 String[] substitutions)
{
    LocalizedMessage localizedMessage =
            new ParameterizedLocalizedMessage(localizationKey,
 substitutions);

    return AccessExceptionContext.create(httpStatus)

 .errorDescription(localizedMessage);
}
```

BEFORE:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 String localizationKey)
{
    return AccessExceptionContext.builder()
                                 .httpStatus(httpStatus)

 .errorDescription(localizationKey)
                                 .errorDescriptionIsKey(true)
                                 .build();
}
```

AFTER:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus,

 String localizationKey)
{
    LocalizedMessage localizedMessage = new
 BasicLocalizedMessage(localizationKey);
    return AccessExceptionContext.create(httpStatus)

 .errorDescription(localizedMessage);
}
```

BEFORE:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus)
{
    return AccessExceptionContext.builder()
                                 .from(httpStatus)

 .httpStatusDescription(httpStatus.getLocalizationKey())

 .httpStatusDescriptionIsKey(true)
                                 .templateFile("template.html")
                                 .contentType("text/html");
}
```

AFTER:

```
private AccessExceptionContext
 createAccessExceptionContext(HttpStatus httpStatus)
{
    return AccessExceptionContext.create(httpStatus)

 .errorDescription(httpStatus.getLocalizedMessage());
}
```

NOTE: this example demonstrates that it is no longer possible to set a template file and its associated content type on an AccessExceptionContext. To generate an error response from a template file, use the TemplateRenderer class. See the JavaDoc for the TemplateRenderer class for more information.

com.pingidentity.pa.sdk.policy.AccessException

The changes to AccessExceptionContext apply to the creation of AccessException because the creation of an AccessException requires an AccessExceptionContext.

In addition to these changes, obtaining information from AccessException has also changed. See the code examples below for more information.

Finally, AccessException no longer derives from IOException and derives directly from Exception instead.

Constructing an AccessException:

BEFORE:

```
private void throwAccessException(String errorDescription,
                                  Throwable throwable) throws
 AccessException
{
    throw new AccessException(errorDescription, throwable);
}
```

AFTER:

```
private void throwAccessException(String errorDescription,
                                  Throwable throwable) throws
 AccessException
{
    LocalizedMessage templateMessaage = new
 FixedMessage(errorDescription);
    throw new
 AccessException(AccessExceptionContext.create(HttpStatus.INTERNAL_SERVER_ERROR)

 .exceptionMessage(errorDescription)

 .cause(throwable)

 .errorDescription(templateMessaage));
}
```

BEFORE:

```
private void throwAccessException(String errorDescription) throws
 AccessException
{
    throw new AccessException(errorDescription);
}
```

AFTER:

```
private void throwAccessException(String errorDescription) throws
 AccessException
{
    LocalizedMessage templateMessage = new
 FixedMessage(errorDescription);
    throw new
 AccessException(AccessExceptionContext.create(HttpStatus.INTERNAL_SERVER_ERROR)

 .exceptionMessage(errorDescription)

 .errorDescription(templateMessage));
}
```

BEFORE:

```
private void createAccessException(int statusCode,
                                   String statusMessage,
                                   String errorDescription)
 throws AccessException
{
    throw new AccessException(statusCode, statusMessage,
 errorDescription);
}
```

AFTER:

```
private void createAccessException(int statusCode,
                                   String statusMessage,
                                   String errorDescription)
 throws AccessException
{
    HttpStatus httpStatus = new HttpStatus(statusCode,
 statusMessage);
    LocalizedMessage templateMessage = new
 FixedMessage(errorDescription);
    throw new
 AccessException(AccessExceptionContext.create(httpStatus)

 .exceptionMessage(errorDescription)

 .errorDescription(templateMessage));
}
```

BEFORE:

```
private void throwAccessException(int statusCode,
                                  String statusMessage,
                                  String errorDescription,
                                  Throwable throwable) throws
 AccessException
{
    throw new AccessException(statusCode, statusMessage,
 errorDescription, throwable);
}
```

AFTER:

```
private void throwAccessException(int statusCode,
                                  String statusMessage,
                                  String errorDescription,
                                  Throwable throwable) throws
 AccessException
{
    HttpStatus httpStatus = new HttpStatus(statusCode,
 statusMessage);
    LocalizedMessage templateMessage = new
 FixedMessage(errorDescription);
    throw new
 AccessException(AccessExceptionContext.create(httpStatus)

 .exceptionMessage(errorDescription)

 .errorDescription(templateMessage)

 .cause(throwable));
}
```

BEFORE:

```
private void throwAccessException() throws AccessException
{
    throw new AccessException(AccessExceptionContext.builder()

 .httpStatusCode(403)

 .httpStatusMessage("Forbidden")
                                                .build());
}
```

AFTER:

```
private void throwAccessException() throws AccessException
{
    throw new
 AccessException(AccessExceptionContext.create(HttpStatus.FORBIDDEN));
}
```

Using the AccessException#getExceptionContext method

BEFORE:

```
AccessExceptionContext context =
 accessException.getExceptionContext();
```

AFTER:

This functionality is no longer supported. The information that used to be provided by the AccessExceptionContext is now provided directly by an AccessException.

Using the AccessException#getHttpStatusCode method

BEFORE:

```
int statusCode = accessException.getHttpStatusCode();
```

AFTER:

```
int statusCode = accessException.getErrorStatus().getCode();
```

Using the AccessException#getHttpStatusMessage method

BEFORE:

```
String statusMessage = accessException.getHttpStatusMessage();
```

AFTER:

```
String statusMessage =
 accessException.getErrorStatus().getMessage();
```

Using the AccessException#setHttpStatusCode method

BEFORE:

```
accessException.setHttpStatusCode(statusCode);
```

AFTER:

This functionality is no longer supported. The status code associated with an AccessException is fixed once it is constructed.

Using the AccessException#setHttpStatusMessage method

BEFORE:

```
accessException.setHttpStatusMessage(statusMessage);
```

AFTER:

This functionality is no longer supported. The status message associated with an AccessException is fixed once it is constructed.

com.pingidentity.pa.sdk.policy.RuleInterceptor

The handleRequest and handleResponse methods on a RuleInterceptor no longer throw an IOException. Instead, they throw an AccessException, which no longer derives from IOException.

Accounting for the RuleInterceptor#handleRequest method signature change

BEFORE:

```
@Override
public Outcome handleRequest(Exchange exchange) throws
 IOException
{
    Outcome outcome = applyPolicy(exchange);

    return outcome;
}
```

AFTER:

```
@Override
public Outcome handleRequest(Exchange exchange) throws
 AccessException
{
    Outcome outcome = applyPolicy(exchange);

    return outcome;
}
```

Account for the RuleInterceptor#handleResponse method signature change

BEFORE:

```
@Override

public void handleResponse(Exchange exchange) throws IOException

{

    applyPolicyToResponse(exchange.getResponse());

}
```

AFTER:

```
@Override
public void handleResponse(Exchange exchange) throws
 AccessException
{
    applyPolicyToResponse(exchange.getResponse());
}
```

com.pingidentity.pa.sdk.policy.error.InternalServerErrorCallback

This class has been removed. Use LocalizedInternalServerErrorCallback instead.

com.pingidentity.pa.sdk.services

com.pingidentity.pa.sdk.services.ServiceFactory

This class is now final and cannot be extended.

com.pingidentity.pa.sdk.siteauthenticator

com.pingidentity.pa.sdk.siteauthenticator.SiteAuthenticatorInterceptor

This interface is no longer a RequestInterceptor or ResponseInterceptor, but it still defines the handleRequest and handleResponse methods:

```
public interface SiteAuthenticatorInterceptor<T extends
 PluginConfiguration>
        extends DescribesUIConfigurable, ConfigurablePlugin<T>
{
    void handleRequest(Exchange exchange) throws AccessException;
    void handleResponse(Exchange exchange) throws
 AccessException;
}
```

Additionally, these methods now only throw an AccessException instead of an IOException or InterruptedException.

Accounting for the SiteAuthenticatorInterceptor#handleRequest method signature change

BEFORE:

```
@Override
public Outcome handleRequest(Exchange exc)
        throws RuntimeException, IOException,
 InterruptedException
{
    // Site authenticator implementation //

    return Outcome.CONTINUE;
}
```

AFTER:

```
@Override
public void handleRequest(Exchange exc) throws AccessException
{
    // Site authenticator implementation //
}
```

Accounting for the SiteAuthenticatorInterceptor#handleResponse method signature change

BEFORE:

```
@Override
public void handleResponse(Exchange exc) throws IOException
{
    // Site authenticator response implementation //
}
```

AFTER:

```
@Override
public void handleResponse(Exchange exc) throws AccessException
{
    // Site authenticator response implementation //
}
```

com.pingidentity.pa.sdk.ui

com.pingidentity.pa.sdk.ui.ConfigurationType

The deprecated PRIVATEKEY enum value has been removed. Instead use a ConfigurationType of ConfigurationType#SELECT and specify the PrivateKeyAccessor.class instance to ConfigurationBuilder#dynamicOptions or UIElement#modelAccessor.

com.pingidentity.pa.sdk.user

com.pingidentity.pa.sdk.user.User

This class has been removed from the SDK. Use the Identity interface instead. An instance of Identity can be retrieved from the Exchange, similar to the User interface.

com.pingidentity.pa.sdk.util

com.pingidentity.pa.sdk.util.TemplateRenderer

The semantics of the renderResponse method have changed so it produces a Response and does not have any side-effects on the specified parameters.

Using the TemplateRenderer#renderResponse method:

BEFORE:

```
private void invokeRenderResponse(TemplateRenderer
 templateRenderer,
                                  Map<String, String> context,
                                  String templateName,
                                  Exchange exchange,
                                  ResponseBuilder builder)
{
    templateRenderer.renderResponse(context, templateName,
 exchange, builder);
}
```

AFTER:

```
private void invokeRenderResponse(TemplateRenderer
 templateRenderer,
                                 Map<String, String> context,
                                 String templateName,
                                 Exchange exchange,
                                 ResponseBuilder builder)
{
    Response response = templateRenderer.renderResponse(exchange,
                                                      context,

 templateName,
                                                      builder);
    exchange.setResponse(response);
}
```

Protect a web-based application using PingFederate and PingAccess in a proxy deployment

Protect a web-based application in a proxy deployment

This document describes the basic steps necessary to protect a web-based application using the combination of **PingAccess** as the access manager and **PingFederate** as the token provider.

Along with a description of the configuration steps that are required, this document includes a functioning example using the **PingAccess QuickStart** application. You can perform the example configuration to achieve a working result and become familiar with this solution, or you can substitute your own data.

Note that this is a basic configuration featuring:

- a PingFederate HTML Form Adapter with an instance of a Simple Password Credential Validator
- the PingFederate Access Token Manager (ATM) using the Internally Managed Reference Token data model

Your configuration may call for additional settings and components to satisfy network and/or security requirements. Links to additional resources that explain these additional settings and components are offered throughout this document.

This document does not detail the usage of identity mappings or authentication requirements that are common components of a typical configuration. As these components are simple to configure and beyond the scope of this document, you can read more about them and other features in the *PingAccess User Interface Reference Guide*.

Prerequisites

To complete the example configuration, the following prerequisites should be satisfied:

1. You have downloaded and performed a basic installation of ***PingAccess 5.1*** and ***PingFederate 9.1*** .
2. You have downloaded the ***PingAccess QuickStart*** **(Login required)** application, unzipped the file, and copied the `\pingaccess-quickstart-5.1.0\pf-dist\PingAccessQuickStart.war` file to `\Program Files\Ping Identity\pingfederate-9.1.0\pingfederate\server\default\deploy`.
3. The PingAccess QuickStart application is hosted and available at `https://mypingfedserver:9031/PingAccessQuickStart`.

To begin, *Configure PingFederate for PingAccess connectivity* on page 378.

Configure PingFederate for PingAccess connectivity

This document provides the sequence of steps for configuring PingFederate components for PingAccess connectivity as required for this solution. In this configuration procedure, you will:

1. *Enable PingFederate roles and protocols* on page 378
2. *Create a password credential validator* on page 378
3. *Configure an IdP adapter* on page 379
4. *Define the scope* on page 379
5. *Create an access token manager* on page 379
6. *Configure an IdP adapter mapping* on page 380
7. *Configure an access token mapping* on page 380
8. *Create an OpenID Connect policy* on page 380
9. *Create a resource server client* on page 381
10. *Create a web session client* on page 381
11. *Create and export a certificate* on page 382

Important:

1. These steps assume you have installed PingFederate. The example assumes that your PingFederate instance is available at `https://mypingfedserver`, using ports `9031` and `9999` respectively for the runtime and administration functions.
2. These steps assume you have installed PingAccess. This example assumes that your PingAccess instance is available at `https://mypingaccessserver` and that `3000` is the default listening port.

Enable PingFederate roles and protocols

Ensure that PingFederate is configured to respond to OAuth and OIDC requests using the following steps. For more information on PingFederate Roles and Protocols, see *Choose roles and protocols*.

1. Navigate to **Server Configuration**# **System Settings**# **Server Settings**.
2. Click **Roles & Protocols**and ensure the following items are selected:
   - **OAuth** (role) and **OpenID Connect** (protocol)
   - **IdP** (role) and **SAML 2.0** (protocol)
3. On the **Federation Info** screen, enter the URL of your PingFederate environment and your SAML 2.0 entity ID. For example:
   - **My Base URL**: `https://mypingfedserver:9031`
   - **SAML 2.0 Entity ID**: `https://mypingfedserver/idp`
4. Click **Save**.

Create a password credential validator

Create a password credential validator, then create a username and password to authenticate with. For more information on Password Credential Validators, see *Manage password credential validators*.

1. Navigate to **Settings**# **Server Configuration**# **Authentication**# **Password Credential Validators**.
2. Click **Create New Instance**.
3. Specify an **Instance Name** and **Instance ID** of your choosing. For example:

```
Instance Name: My_PCV
Instance ID: mypcv
```

4. In the **Type** list, select **Simple Username Password Credential Validator**.
5. On the **Instance Configuration** page, click **Add a new row to 'Users'**.
6. Create a **Username**, then create and confirm a **Password**.

7. Click **Update**.
8. On the **Summary** page, click **Done**.
9. Click **Save**.

Configure an IdP adapter

Configure an IdP adapter to look up session information and provide user identification to PingFederate. This example uses an instance of the HTML Form Adapter with an instance of the Simple Password Credential Validator. For more information, see *Manage IdP adapters*.

1. Navigate to **Identity Provider# Application Integration# Adapters**.
2. Click **Create New Instance**.
3. Specify an **Instance Name** and **Instance ID** of your choosing. For example:

```
Instance Name: My_IdP
Instance ID: myidp
```

4. In the **Type** list, select **HTML Form IdP Adapter**.
5. On the **IdP Adapter** page, click **Add a new row to 'Credential Validators'.**
6. In the **Password Credential Validator Instance** list, select the password credential validator you created (*Example:* `My_PCV`) and click **Update**.
7. On the **Adapter Attributes** page, next to the `username` attribute, click to select the **Pseudonym** checkbox
8. On the **Summary** page, click **Done**.
9. Click **Save**.

Define the scope

Use the Scope Management screen to define the default scope. For more information, see *Define scopes*.

1. Navigate to **OAuth Server# Authorization Server# Scope Management**.
2. On the **Common Scopes** page, enter the following scope values and their descriptions one at a time, clicking **Add** with each entry:

Scope Value	Scope Description
address	address
email	email
openid	openid
phone	phone
profile	profile

3. On the **Default Scope** page, enter a description, for example: `default scope`.
4. Click **Save**.

Create an access token manager

Create an access token that is used to grant access and control access parameters. This sample configuration uses an instance of the Access Token Manager (ATM) using the Internally Managed Reference Tokens data model. For more information, see *OAuth access token management*.

1. Navigate to **OAuth Server# Token Mapping# Access Token Management**.
2. Click **Create New Instance**.
3. Specify an **Instance Name** and **Instance ID** of your choosing. For example:

```
Instance Name: General Access Token
Instance ID: GeneralAccessToken
```

4. In the **Type** list, select **Internally Managed Reference Tokens**.
5. On the **Access Token Attribute Contract** page, in the **Extend the Contract** field, enter `UserName` and click **Add**.
6. On the **Summary** page, click **Save**.

Configure an IdP adapter mapping

Configure an IdP adapter mapping to map attributes. For more information, see *Manage IdP adapter mappings for OAuth*.

1. Navigate to **OAuth Server**# **Grant Mapping**# **IdP Adapter Mapping**.
2. In the **Source Adapter Instance** list, select the adapter you created.
3. Click **Add Mapping**.
4. On the **Contract Fulfillment** page, in the **Source** lists, select **Adapter** for both USER_KEY and USER_NAME.
5. In the **Value** lists, select `username` for both items.
6. On the **Summary** page, click **Save**.

Configure an access token mapping

Configure an access token mapping to map attributes to be requested from the OAuth resource server with the access token. For more information, see *Manage access token mappings*.

1. Navigate to **OAuth Server**# **Token Mapping**# **Access Token Mapping**.
2. In the **Context** list, select `Default` or select your IdP adapter instance.
3. In the **Access Token Manager** list, select the access token you created. For example:

```
GeneralAccessToken
```

4. Click **Add Mapping**.
5. On the **Contract Fulfillment** page, in the **Source** list, select `Persistent Grant`.
6. In the **Value** list, select `USER_KEY`.
7. On the **Summary** page, click **Save**.

Create an OpenID Connect policy

Configure an OpenID Connect policy so you can define OpenID Connect policies for client access to attributes mapped according to OpenID specifications. For more information, see *Configure OpenID Connect policies*.

1. Navigate to **OAuth Server**# **Token Mapping**# **OpenID Connect Policy Management**.
2. Click **Add Policy**.
3. Specify a **Policy ID**, for example: `OIDC`
4. Specify a **Name**, for example: `OIDC`
5. In the **Access Token Manager** list, select the access token you created. For example:

```
GeneralAccessToken
```

6. On the **Attribute Contract** page, delete all items beneath **Extend the Contract**.
7. On the **Contract Fulfillment** page, in the **Source** list, select `Access Token`.
8. In the **Value** list, select `UserName`.
9. On the **Summary** page, click **Done**.
10. Click **Save**.
11. Beside the policy you created, click **Set as Default**.
12. Click **Save**.

Create a resource server client

Configure an OAuth client for use with PingFederate token provider resource server configuration in PingAccess. For more information, see *Manage OAuth clients*.

1. Navigate to **OAuth Server# Clients# Manage All**.
2. Click **Add Client**.
3. Specify a **Client ID**. For example:

   ```
   pa_rs
   ```
4. Specify a **Name**. For example:

   ```
   PingAccessResourceServer
   ```
5. Select **Client Secret**.
6. Generate a secret by clicking **Generate Secret**. Copy this secret to a secure location so that you can use it in PingAccess configuration.
7. In the **Redirect URIs** field, add the OIDC callback redirect to the PingAccess server. For example:

   ```
   https://mypingaccessserver:3000/pa/oidc/cb
   ```
8. Click **Add**.
9. For the **Allowed Grant Type** setting, select the **Access Token Validation (Client is a Resource Server)** check box.
10. Click **Save**.
11. Click **Save**.

Create a web session client

Configure an OAuth client for use with web session configuration in PingAccess. For more information, see *Manage OAuth clients*.

1. Navigate to **OAuth Server# Clients# Manage All**.
2. Click **Add Client**.
3. Specify a **Client ID**, for example:

   ```
   pa_wam
   ```
4. Specify a **Name**. For example:

   ```
   PingAccessWebAccessManagement
   ```
5. Select **Client Secret**.
6. Generate a secret by clicking **Generate Secret**. Copy this secret to a secure location so that you can use it in PingAccess configuration.
7. In the **Redirect URIs** field, add the OIDC callback redirect to the PingAccess server, for example:

   ```
   https://mypingaccessserver:3000/pa/oidc/cb
   ```
8. Click **Add**.
9. Select the **Bypass Authorization Approval** check box.
10. For the **Allowed Grant Type** setting, select the **Authorization Code** check box.
11. Click **Save**.
12. Click **Save**.

Create and export a certificate

Create and export a certificate for the PingFederate server that you will import to PingAccess to establish trust. For more information, see *Manage SSL server certificates*.

1. Navigate to **Server Configuration# Certificate Management# SSL Server Certificates**
2. Click **Create New**.
3. In the **Common Name** field, enter the PingFederate server address. For example:

   ```
   mypingfedserver
   ```
4. Complete the remaining fields as required.
5. Click **Next**
6. Click **Done**.
7. Under the **Action** heading, click **Activate for Runtime Server**.
8. **Export** the certificate only.
9. Click **Save**.

Next, *Connect to PingFederate and configure an application in PingAccess* on page 382.

Connect to PingFederate and configure an application in PingAccess

This document provides the sequence of steps for configuring the PingAccess components required for this solution. In this configuration procedure, you will:

1. *Import certificates and create a trusted certificate group* on page 382
2. *Configure the token provider* on page 383
3. *Create a web session* on page 383
4. *Create a virtual host* on page 384
5. *Create a site* on page 384
6. *Create an application* on page 384

Important: The examples in this procedure assume that the PingAccess QuickStart application is available at the following address:

```
https://mypingfedserver:9031/PingAccessQuickStart
```

Import certificates and create a trusted certificate group

Import a certificate for the PingFederate server to establish trust. For more information, see *Certificates*.

1. Navigate to **Settings# Security# Certificates**.
2. Click **+** to the right of the **Certificates** subheading.
3. Enter an **Alias** for the certificate. For example:

   ```
   PingFed
   ```
4. Click **Choose File** to select the certificate.
5. Click **Add** to import the certificate. A new certificate row appears on the Certificates page.
6. Click **+** to the right of the **Trusted Certificate Groups** heading.
7. Drag a certificate onto the box that appears.
8. Enter a **Name** for the group in the box that appears. For example:

   ```
   PingFed
   ```
9. Click **Add**.

Configure the token provider

Establish communication with the token provider, PingFederate. For more information, see *Manage Token Provider*.

1. Navigate to **Settings# System# Token Provider# Runtime**.
2. Enter the **Host** name or IP address for the PingFederate Runtime. For example:

   ```
   mypingfedserver
   ```
3. Enter the **Port** number for PingFederate Runtime. For example:

   ```
   9031
   ```
4. Select **Secure**.
5. From the **Trusted Certificate Group** list, select the `PingFed` certificate group.
6. Click **Save**.
7. Navigate to **Settings# System# Token Provider# Administration**.
8. Enter the **Host** name or IP address for access to the PingFederate Administrative API. For example:

   ```
   mypingfedserver
   ```
9. Enter the **Port** number for access to the PingFederate Administrative API. For example:

   ```
   9999
   ```
10. Enter the PingFederate **Admin Username**.
11. Enter the **Admin Password**.
12. Select **Secure**.
13. From the **Trusted Certificate Group** list, select the `PingFed` certificate group.
14. Click **Save**.
15. Navigate to **Settings# System# Token Provider# OAuth Resource Server**
16. Enter the OAuth **Client ID** you defined when creating the PingAccess OAuth client in PingFederate. For example:

    ```
    pa_rs
    ```
17. Enter the **Client Secret** you defined when creating the PingAccess OAuth client within PingFederate.
18. In the **Subject Attribute Name** field, enter the attribute you want to use from the OAuth access token as the subject for auditing purposes. For example:

    ```
    username
    ```
19. Click **Save**.

Create a web session

Create a web session to control access. For more information, see *Web Sessions*.

1. Navigate to **Settings# Access# Web Sessions**
2. On the Web Session page, click **Add Web Session**.
3. Enter a unique **Name** for the web session, up to 64 characters, including special characters and spaces. For example:

   ```
   PF_WAM
   ```
4. Select `Encrypted JWT` for **Cookie Type**.

5. Specify the **Audience** that the PA Token is applicable to, represented as a short, unique identifier between 1 and 32 characters. For example:

   ```
   global
   ```
6. Specify the **OpenID Connect Login Type** `CODE`.
7. Specify the **Client ID**. For example:

   ```
   pa_wam
   ```
8. Specify the **Client Secret**
9. Click **Save**.

Create a virtual host

Create a virtual host that PingAccess will respond to. For more information, see *Virtual Hosts*.

1. Navigate to **Settings**# **Access**# **Virtual Hosts**.
2. Click **Add Virtual Host**.
3. Enter the **Host** name for the Virtual Host. For example:

   ```
   mypingaccessserver
   ```
4. Enter the **Port** number for the Virtual Host. For example:

   ```
   3000
   ```
5. Click **Save**.

Create a site

Create a site to define the location of the application that PingAccess is protecting. For more information, see *Sites*.

1. Navigate to **Main**# **Sites**# **Sites**.
2. Click **Add Site**.
3. Specify a **Name**. For example:

   ```
   QuickStart
   ```
4. Specify the **Target**. The format for this is `hostname:port`. For example:

   ```
   mypingfedserver:9031
   ```
5. Select **Secure** and choose the `PingFed` **Trusted Certificate Group**.
6. Click **Save**.

   Note: If the target site cannot be contacted, the site is saved and a warning is displayed indicating the reason the site was not reachable.

Create an application

Create an application to define the resource that PingAccess is protecting. For more information, see *Applications*.

1. Navigate to **Main**# **Applications**.
2. Click **Add Application**.

3. Provide a unique name for the application. For example:

   ```
   QuickStart
   ```
4. Specify the context at which the application is accessed at the site. For example:

   ```
   /PingAccessQuickStart
   ```
5. Specify the virtual host for the application. For example:

   ```
   mypingaccessserver:3000
   ```
6. Specify the application type `Web` and select the **Web Session** for the application. For example:

   ```
   PF_WAM
   ```
7. Specify the application destination type Site and select the **Site** requests are sent to when access is granted. For example:

   ```
   PingAccessQuickStart
   ```
8. Select the **Require HTTPS** option.
9. Select the **Enabled** checkbox.
10. Click **Save**.

Test the configuration

To test the solution after configuration is complete:

1. Open your browser.
2. Navigate to the application using a combination of the Virtual Host and Context Root that you defined in PingAccess. For example:

   ```
   https://mypingaccessserver:3000/PingAccessQuickStart
   ```
3. Login using the credentials you created for the PingFederate *password credential validator*.

Protect applications using PingAccess and PingOne for Customers

Solution overview

This document provides the steps required to configure PingAccess as part of the use case to provide secure external access to applications using PingAccess and PingOne for Customers. In this scenario, PingAccess provides an external path to applications while PingOne for Customers acts as the token provider for associated sessions.

This solution requires you to perform the following tasks. For more information about the requirements and options available for each task, review the task.

- *Configure PingAccess to use PingOne for Customers as the token provider*
- *Configure a PingAccess application* for each application you want to protect and make available as part of this solution. Applications may require configuration of:
  - A virtual host
  - A web session or access token validator
  - A site
  - An application

When the configuration is complete, you can test the application using the **Virtual Host** and **Context Root** that you assign to it in PingAccess.

Configure PingAccess to use PingOne for Customers as the token provider

ⓘ **Tip:** For more information on configuring the token provider, see *Configure a token provider*.

Assumptions

- You have installed PingAccess and can access the *Administrative console*. For information on installing PingAccess, see *Install PingAccess*.

ⓘ **Note:** The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:

```
Username: Administrator
Password: 2Access
```

- You have *configured an application* in PingOne for Customers.

Configure the token provider

1. Navigate to **Settings**# **System**# **Token Provider**.
2. Select the **PingOne** token provider type.
3. In the **Issuer** field, enter the PingOne for Customers **Issuer** URL. To obtain the Issuer URL from PingOne for Customers, in PingOne for Customers, navigate to the **Configuration** tab of an application and copy the **Issuer** value.
4. Provide a **Description** of the token provider.
5. In the **Trusted Certificate Group** list, select **Java Trust Store** or **Trust Any**.
6. Click **Save**.

Configure a PingAccess application

This document describes the steps required to configure PingAccess applications. Use these instructions for each application that you want to configure.

Assumptions:

- You have installed PingAccess and can access the *Administrative console*. For information on installing PingAccess, see *Install PingAccess*.

ⓘ **Note:** The default credential set should be changed upon first usage. The default credentials for your PingAccess installation are:

```
Username: Administrator
Password: 2Access
```

- You have *configured an application* in PingOne for Customers.
- PingAccess is *configured* to use PingOne for Customers as the token provider.

Create a virtual host

ⓘ **Tip:** For more information on creating a virtual host, see *Create a virtual host*.

1. Navigate to **Settings**# **Access**# **Virtual Hosts**.
2. Click **Add Virtual Host**.
3. In the **Host** field, enter the Host name you want to use for the Virtual Host.
4. In the **Port** field, enter the port number you want to use for this virtual host..

5. Click **Save**.

Create a web session

ⓘ **Tip:** For more information on creating a web session, see *Create a web session*.

ⓘ **Note:** A web session is only used when protecting a web application. To protect APIs, you will configure an Access Token Validator.

1. Navigate to **Settings**# **Access**# **Web Sessions**.
2. Click **Add Web Session**.
3. Provide a **Name** for the web session.
4. Select the **Cookie Type**, either **Signed JWT** or **Encrypted JWT**.
5. Provide a unique value for the **Audience**.
6. In the **Client ID** field, enter the PingOne for Customers **Client ID**. The Client ID can be found on the **Profile** tab of the application you created.
7. In the **Client Secret** field, enter the **Client Secret** found on the application's **Configuration** tab.
8. Ensure the **Scopes** you specify match those configured for the PingOne for Customers application. Scopes are found on the **Access** tab of your PingOne for Customers application.
9. Click **Save**.

Create a site

ⓘ **Tip:** For more information on creating a site, see *Manage Sites*.

ⓘ **Note:** In some configurations, it is possible that a site may contain more than one application. A site can be used with more than one application, where appropriate.

1. Navigate to **Main**# **Sites**# **Sites**.
2. Click **Add Site**.
3. Specify a **Name** for the site.
4. Enter the site **Target**. The target is the hostname:port pair for the server hosting the application. Do not enter the path for the application in this field. For example, an application at `https://mysite:9999/AppName` will have a target value of `mysite:9999`
5. Indicate whether or not the target is expecting **Secure** connections.
6. If the target is expecting secure connections, set the **Trusted Certificate Group** to **Trust Any**.
7. Click **Save**.

Create an application

You will create an application in PingAccess for each application that you want to protect.

ⓘ **Tip:** For more information on creating an application, see *Manage Applications*.

1. Navigate to **Main**# **Applications**.
2. Click **Add Application**.
3. Specify a **Name** for the application.
4. Optionally, enter a **Description** for the application.
5. Specify the **Context Root** for the application. For example, an application at `https://mysite:9999/AppName` will have a context root of `/AppName`. If the application is on the root of the server, you can

set the context root as `/`. The context root must begin with a slash (/), must not end with a slash (/), and can be more than one layer deep, for example, `/Apps/MyApp`.

6. Select the **Virtual Host** you created.

> ⓘ **Note:** The combination of Virtual Host and Context Root must be unique in PingAccess.

7. Specify **Application Type** of **Web**.
8. Select the **Web Session** you created.
9. Select the **Site** you created that contains the application.
10. Select **Enabled** to enable the site when you save.
11. Click **Save**.

Migrating Access Management Infrastructure to PingAccess

Introduction

Choose from one of the following topics:

- *About PingAccess Policy Migration (PAPM)* on page 388
- *Platform architecture* on page 388
- *PAPM naming process* on page 390

About PingAccess Policy Migration (PAPM)

PingAccess Policy Migration simplifies Access Management infrastructure migration from CA SSO (Siteminder) and Oracle Access Manager (OAM) to PingAccess. PAPM enables the easy migration of policy and configuration from the existing system to PingAccess, allowing for easier configuration of PingAccess applications and policies while keeping existing policy behavior.

Platform architecture

An embedded Jetty server powers the PAPM platform. The applications found in the User Interface leverage a local database. It features:

- **Single Deployment Directory**: All deployment code and the included database are packaged in a single directory. This enables a simple deployment package and reusable configurations.
- **Login Flexibility:** Login to the PAPM platform is available via a file-based login or external LDAP with group-based authorization.
- **Backup:** PAPM is not designed to be highly available. The database is regularly archived in a local backup folder. However the database should be periodically backed up onto another server for disaster recovery.

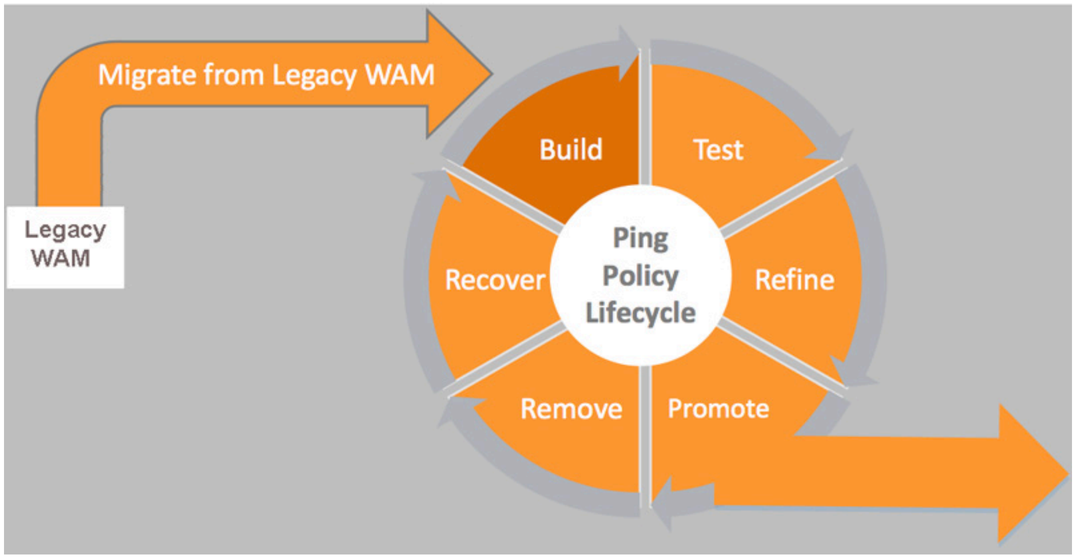
> ⓘ **Note:** PAPM requires API access to the PingAccess and PingFederate Admin API consoles for EVERY available environment

Automating policy management

Successful policy automation begins with policy creation. The PAPM methodology builds policies that are secure, portable, flexible, and well named with policy automation.

Policy Management Lifecycle

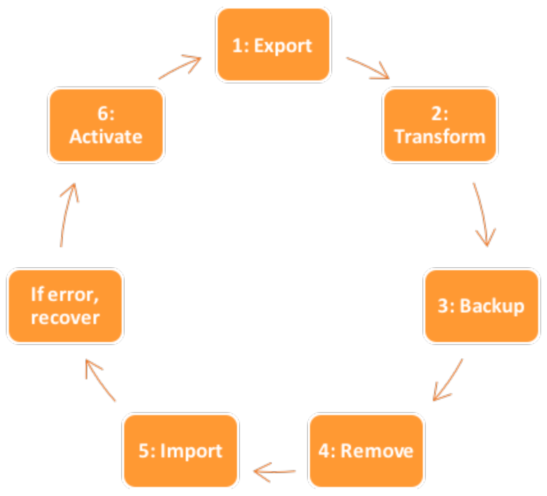
Before embarking on a large-scale policy management project, it is important to understand the lifecycle of a Ping Policy and the issues that limit successful operationalization and automation:



1. **Build** - The policy lifecycle starts with creating application policies. This is the most critical stage of policy development and often the predecessor to an automation program's success or failure.
2. **Test** - New policies test for logic, SSO integration, and performance. The applications protected by PingAccess policies deploy on the network in production before they are tested. This creates an unnecessary linkage between the SSO, operations, and application teams. In production, debugging activities are limited to late night downtime that SSO teams must accommodate to execute policy validation.
3. **Refine** - Correcting issues discovered during policy testing typically requires a policy change.
4. **Promote (and Policy Promotion Process)** - Once a policy is tested and ready for implementation, it promotes to a production environment. PAPM automates the Policy Migration Process for reliable / error free migrations. It reduces the pitfalls that can occur when the administrator manually inputs policy data for policy migration.

   PingAccess and PingFederate leverage APIs to export and import policy data for migration. However, they do not support the changes for multiple environments or for singular application policies. For example, the virtual hostname for an application in Dev (e.g. app-dev.company.com) will always be different than the virtual hostname in production (e.g. app.company.com). Automation requires any environment-related mutable data be managed. It is also critical to migrate data changes and any shared secrets that PingAccess and PingFederate are unable to export.

The Policy Promotion Process



- **1. Package(Export):** Export an application policy and all of its dependencies. This package is stored in the H2 database.
- **2. Transform**: Change all environment dependent data to the target environment. (E.g. map from dev.company.com to sit.company.com).
- **3. Backup**: Prior to importing the new policy, create a backup of the existing application policy and all of its dependencies.
- **4. Remove**: Remove the existing application policy and its dependencies.
- **5. Import**: Create a new application from the transformed export.
  - **If Error, Recover:** Use the backup to recover from any errors.
- **6. Activate**: Once the policy is pushed in PingAccess, it becomes active. The push to PingFederate must be separately pushed to the cluster. The PAPM will automatically call the API to push to the cluster and fully activate the application policy.

> **Note:** It is also possible to merge and link application policies instead of the simpler approach of remove and replace. An organization should standardize its approach for consistent policy management.

5. **Remove** - Administrators are required to remove unwanted Ping policies meticulously in the reverse order created. Removing a top-level application policy also disconnects the links to policy objects (which may lead to policy orphans).
6. **Recover** - While no change is risk-free, the ability to recover policy information from a backup or previous version is helpful. Because the PingAccess Admin stores policy data in a flat-file database, it is impossible to recover individual copies of application policies.

PAPM naming process

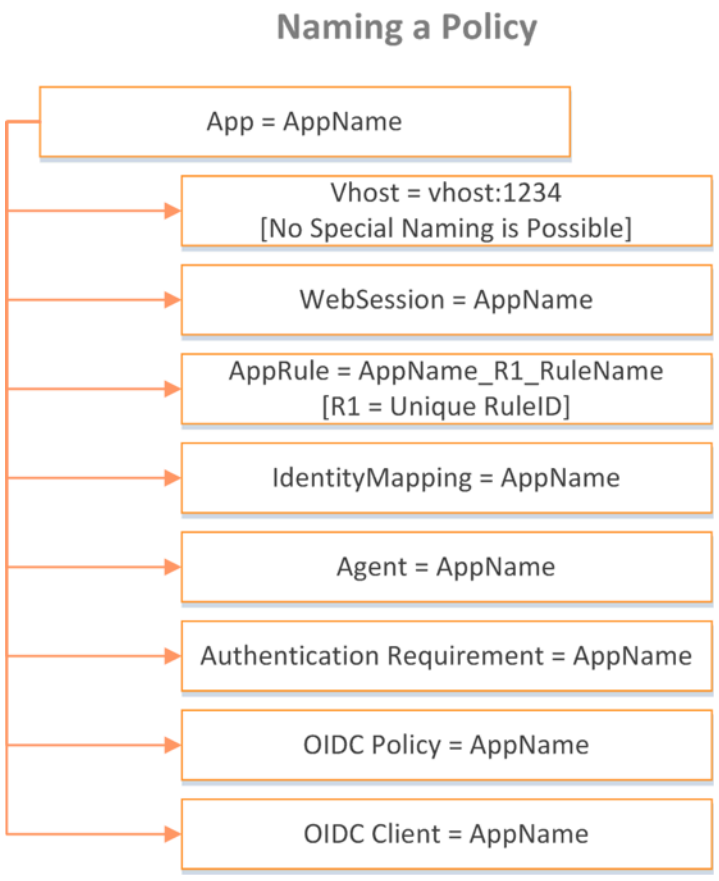
Administrators may find themselves creating poorly named policies that lack scalability and portability in an attempt to maintain legacy naming conventions or policy configurations. An administrator can choose to name policy objects without restraint or adherence to these standards. Reused policy objects can lead to a number of issues, such as accidentally changing a cross-dependent policy object with unintentional impacts to an application.

Standardized names

With standardized names, it is easy to identify the links between an application policy, the HTTP headers it outputs, the web session, the OIDC policy and OIDC client just by looking at the name, even when the policy is in both PingAccess and PingFederate. Naming standards replace the effort it takes to manually setup these relationships using both PingAccess and PingFederate.

Naming a policy

Policy objects are built using the application policy name (e.g. "MyTestPolicy" or "AppName" as shown in diagram). Policy objects are immediately recognizable via their name. This is also important when cleaning up policy object orphans that link to an application policy. This naming structure is baked into PAPM.



PingAccess Policy Migration

PingAccess Policy Migration (PAPM) v1.0 delivers the following:

- *Settings* on page 391
- *Policy Migration for CA SSO (SiteMinder)* on page 391
- *Policy Migration for Oracle Access Manager 10g/11g* on page 392

Settings

PAPM application settings, such as those for users and templates, are configured in one place using **Settings**.

Policy Migration for CA SSO (SiteMinder)

PingAccess customers migrate the policies created using the legacy WAM provider CA SSO (formerly SiteMinder) with Policy Migration for CA SSO. CA SSO policies are created using three distinct methods that are substantially different from the Ping policy structure. This increases the complexity of migration to Ping.

A typical migration involves:

- New infrastructure deployment
- Integration of PingFederate / PingAccess with the traditional WAM solution
- A complex migration of application policy data that has accumulated over the course of a decade or more.

Policy Migration for CA SSO automates the migration of policies from CA SSO to PingAccess and PingFederate. It creates well-named, standalone, and portable PingFederate and PingAccess policies using data exported from SiteMinder. It can eliminate migration complexities,eliminate time-consuming UI searches, and reduce the errors of manually keyed data.

Features:

- **SiteMinder Exports**: Import CA SSO (SiteMinder) XPS export files.
- **JSON-based Format:** Transformation of XPS data into a generic, JSON-based format for easy data manipulation
- **Fast Transformation**: High-speed XML transformation using a background database ETL process.
- **Easy Policy Views:** Single page view of policy data for user convenience. All SiteMinder policy data is compiled on a single page and matched with PingAccess data.
- **Customizable Mapping:** Automatic mapping of policy data into PingAccess objects - with manual manipulation, if required.
- **Auto Build**: Automatic creation of PingFederate OIDC objects including LDAP attribute lookups.

Policy Migration for Oracle Access Manager 10g/11g

PingAccess Policy Migration for Oracle Access Manager (OAM) automates the process of exporting OAM authentication and authorization policies to migrate them to PingAccess and PingFederate. PAPM simplifies the migration of OAM policies to PingAccess and PingFederate by exporting the policy domains from both versions to a common JSON format that PAPM can understand. This process helps remove the guesswork of how your application authentication and authorization policies map to the PingAccess and PingFederate format.

Policy Migration for OAM provides three key components to enable the policy migration:

- **An Export and Normalization Module**: This utility can pull policies from OAM 10g or 11g and normalizes the export into a unified format.
- **The Policy Migration UI**: It provides a logical interface to quickly review the components of an application policy before it is imported into the Ping environment.
- **The PAPM OAM Rule Evaluator**: OAM policies often contain highly complex URL matching patterns (a combination of regex and special characters) and policy logic. The PAPM OAM Rule Evaluator loads directly into the PingAccess runtime engine to perform the same URL parsing and policy logic available in OAM. Since this is a plugin authorization engine, it supports policy caching as well.

Together, these components enable secure policy migrations with minimal user interaction and vastly reduce the time involved with the overall migration to the Ping platform.

Installation and setup

Choose from one of the following topics:

- *Installation* on page 392
- *Setup* on page 395

Installation

The installation process consists of the following high-level activities:

System requirements on page 393

Test connectivity on page 393

Deploy the PAPM platform on page 393

Log in to PAPM on page 394

Import license on page 395

System requirements

The following prerequisites are required to install the application:

- PingAccess 5.0+
- PingFederate 9.0+
- Oracle Java SE Runtime Environment (Server JRE or JDK) 8 (64-bit)
- System Environment Variable JAVA_HOME must exist and be set to a value that represents the location of your Java installation (ie; C:\Program Files\Java\jre_1.8.0_191).

  > **Note:** When adding the environment variable entry via the Windows UI, do not place quotation marks around the path.

- Internet Explorer 11 or above, Microsoft Edge, Mozilla Firefox or Google Chrome
- Valid license for PAPM

> **Note:** For optimal user experience, PAPM requires a minimum browser window size of 1280px x 1024px.

Test connectivity

Before setting up the platform, test connectivity to PingAccess, PingFederate, or both, as needed using existing software. The PingAccess and PingFederate servers must be running to configure PAPM. Make sure the PingAccess server connects to PingFederate via OIDC connection.

Deploy the PAPM platform

About this task

This document describes PAPM installation for Unix and Windows-based systems. Support for running PAPM is provided via shell scripting. There is currently no functionality that allows the product to run as a service.

Prior to starting the installation, the following prerequisites must be satisfied:

- The 64-bit Oracle JRE/JDK must be installed.
- System Environment Variable JAVA_HOME must exist and be set to a value that represents the location of your Java installation (ie; C:\Program Files\Java\jre_1.8.0_191).

  > **Note:** When adding the environment variable entry via the Windows UI, do not place quotation marks around the path.

- You must have a valid PingAccess Policy Migration license file.
- A PKCS 12 file must be created with the keypair. See *Step 3*.

Steps

1. Download the distribution file.
2. Extract the distribution file contents into your Ping Identity installation directory.

3. Add or create a PKCS12 file that contains the PAPM keypair.

   Note: To generate a self signed certificate in Unix that works with the default configuration, run the following command from the PAPM home directory:

   ```
   keytool -genkeypair -alias papm -keypass 2Federate -storepass
    2Federate -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore
    config/keystore.p12 -validity 365 -dname CN=localhost,C=US
   ```

   The file name and location, alias, and credentials can be customized by editing `<PAPM_HOME>/config/application.properties`.
4. In a command prompt or terminal window, change to the PingAccess Policy Migration bin directory:
   - On Linux: `cd <PAPM_HOME>/bin`
   - On Windows: `cd <PAPM_HOME>\bin`
5. Initiate the run script for the platform:
   - On Linux: `./run.sh`
   - On Windows: `run.bat`
6. Secure the PAPM login by configuring the following properties in `<PAPM_HOME>/config/application.properties` as appropriate:

Property	Description
`papm.account-locking.login-failure-threshold`	The maximum number of consecutive failed login attempts before a user or IP address is locked out. If this property is set to 0, lockout is disabled.
`papm.account-locking.lockout-period-minutes`	Specifies the duration of the lockout period in minutes. If this property is set to 0, lockout is disabled.
`papm.account-locking.ip-address-source.header-names[<value>]`	Specifies a header name to search for in the list, where `<value>` represents the value's location in the list order. Multiple values can be defined (For example, `[0]`, `[1]`, `[2]`).
`papm.account-locking.ip-address-source.list-value-location`	Specifies whether, when a list of values is in the header, the first value or the last value in the list should be used as the IP Source value. The default value is `Last`.
`papm.account-locking.ip-address-source.fall-back-to-last-hop-ip`	Specify `true` to indicate that if none of the listed headers is present in the request, the upstream IP address should be used for rule evaluation. If this value is disabled (`false`) and no headers match, the network range rule will return a `Forbidden` status.

Log in to PAPM

About this task

Note: For optimal user experience, PAPM requires a minimum browser window size of 1280px x 1024px.

Steps

1. Open a web browser and type the PAPM URL:

   ```
   https://<servername>:9011
   ```

   Note: The port may be customized by editing <PAPM_HOME>/config/application.properties.
2. Select and import the PAPM license.
3. Log in using the default credentials: `administrator/2Federate`.

Import license

About this task

PAPM requires a valid license to function. If a valid license is not found, you will be asked to import one. To import a license directly into the tool:

Steps

1. Using the Settings tool, click **License Management** from the left navigation menu.
2. Click the **Select License File** button.
3. Navigate to the license key file location and select the license key file. The path and file name of the license key file display and the license key validation process executes. The license key is stored on the file system: `<PAPM_HOME>/config/papm.lic`
4. Once the key validates, click **Update License** to update the license key.
5. To complete the license key process, log out of the PAPM and then log back in.

Tip: To manually update the license:

a. Rename the updated license file to `papm.lic`.
b. Use this file to replace the existing license at `<PAPM_HOME>/config`.
c. Log out of PAPM, then log back in.

Setup

Once installed, additional setup procedures may be required prior to using PAPM. These are typically one-time setup activities but some may require multiple iterations (such as setting up environments).

- ***Change login type*** **on page 400** : The default type is initially set to File Login. Typically, PAPM users prefer to use LDAP System Account Login.
- ***Change the database password*** (Optional): Change the database password for increased security.
- ***Create Environments*** **:** Create an environment in PAPM for each Ping environment.

Upgrade PingAccess Policy Migration

Before you begin

This document provides instructions for upgrading a standalone instance of PingAccess Policy Migration.

To run the upgrade utility, you will need the following:

- The PingAccess Policy Migration Upgrade Utility archive
- The PingAccess Policy Migration distribution zip file

Copy the required files to the system being upgraded, and unpack the PingAccess Policy Migration Upgrade Utility archive.

Any warnings or errors encountered are recorded in `log/upgrade.log`, as well as on the screen while the utility is being run.

About this task

Use the PingAccess Policy Migration Upgrade Utility to upgrade PingAccess Policy Migration to a more recent version.

> ⓘ **Important:** Before you upgrade, create a backup of your existing PingAccess Policy Migration configuration. In the event of an upgrade failure, you can restore the backup configuration.

Steps

1. Stop the running PingAccess Policy Migration instance.
2. Unpack the upgrade utility zip file.
3. Change to the upgrade utility's `bin` directory.
4. Run the PingAccess Policy Migration Upgrade Utility:

   - On Windows: `run.bat` *`<sourcePAPMRootDir> <outputDir> <papmZip>`*
   - On Linux: `./run.sh` *`<sourcePAPMRootDir> <outputDir> <papmZip>`*

   For example: `./run.sh pingaccess-policy-migration-1.0 pingaccess-policy-migration-1.0.1 pingaccess-policy-migration-1.0.1.zip`

   Parameter Definitions

   The command-line parameters are the same regardless of the platform, and are defined as follows:

Parameter	Value description
<<sourcePAPMRootDir>>	The home directory for the source PingAccess Policy Migration version
<outputDir>	The target directory which will contain the unpacked PingAccess Policy Migration distribution
<<papmZip>>	The PingAccess Policy Migration distribution for the target version

   > ⓘ **Note:** In the context of an upgrade, "source" refers to the old version of PingAccess Policy Migration, and "target" refers to the new version.

5. *Start* the new PingAccess Policy Migration instance.

Settings

PAPM application settings, such as those for users, licenses and templates, are configured in one place using **Settings**.

Environment management

Configure the connection parameters to each of the target PingAccess and PingFederate environments from the **Environment Management** screen. PAPM uses these environment settings to perform all of the policy administration allowed by the tool for each of the paired PingAccess and PingFederate deployments you wish to manage via PAPM.

Note: The environments created here are also used for the tracking of audit and policy recovery data. The environment name (for example, dev) is linked to this data. In this version, it is not possible to rename or delete an environment. An environment can be disabled and will disappear in the rest of the UI.

Important: If your environment requires the use of TLS Server Name Indication (SNI), use fully qualified domain names when entering the PingAccess and PingFederate host values. SNI is not supported when using hostnames (e.g. localhost) or IP addresses.

Important: If your PingAccess or PingFederate admin consoles are behind a reverse proxy and mapped to a path other than /, PingAccess Policy Migration will not be able to connect, as this version of PingAccess Policy Migration uses specific API paths.

Environment parameter	Example value
Hierarchy Index: Numerical index to prioritize the drop down list of environments.	1
Environment: The environment name in all lower case. There must be no leading or trailing spaces.	dev
PA API Version: Version number of the PingAccess API REST URL. It is also linked to the set of Java data beans that carry the compatible versioning structure to allow for backwards and forwards version pushes. PingAccess 3 = v1 API PingAccess 4 = v2 API PingAccess 5 = v3 API	v3
PA Agent Listener [<Host>:<Port>]: This is the agent listener hostname and port that is used to create PingAccess Agents and their associated `bootstrap.properties`.	mypaserver:3030
PA Engine Listener [<Host>:<Port>]: The engine listener hostname and port to connect to the PingAccess Proxy. This is used by the PingAccess Test Tool to create PingAccess engines and their associated `bootstrap.properties`.	mypaserver:3000
PA Admin Host [<Host>]: Hostname of the PingAccess administrative host node (or VIP) used to connect to the PingAccess Administrative API to allow PAPM to read and update configuration used for policy administration and migration use cases..	mypaserver
PA Admin Port [<Port>]: PingAccess administrative host port.	9000
PA LoginType: Basic authentication (username / password) or API authentication to the PingAccess admin APIs.	Basic
PA Username: Administrative username for PingAccess.	Administrator

PA Password: PingAccess admin password. ⓘ **Note:** To change the password when updating an existing environment, clear the password and enter a new value. Otherwise, leave the password unchanged.	mypapassword This field is rendered as masked values (*).
API Scope: PingAccess API login scope (only used for API-based authentication).	Admin
API Granttype: PingAccess API Grant Type used for API authentication.	client_credentials
API ClientID: PingAccess OIDC API Client ID configured in PingAccess and PingFederate.	admin_api_client
API Client Secret: PingAccess API Client Secret that was configured for the OIDC client.	myapipassword This field is rendered as masked values (*).
API TokenURL[<Host>:<Port>/<URL>]: The PingFederate API Token Login URL.	https://pfserver:9031/as/token.oauth2
PF Admin Host [<Host>]: PingFederate administrative host node (or VIP).	mypfserver
PF Admin Port [<Port>]: PingFederate administrative host port.	9999
PF Username: An administrator login name with access to the PingFederate API's.	Administrator
PF Password: The PingFederate administrative password for the user configured above. ⓘ **Note:** To change the password when updating an existing environment, clear the password and enter a new value. Otherwise, leave the password unchanged.	mypfpassword This field is rendered as masked values (*).

Create a new environment

About this task

This section contains information about setting up environments in PingFederate and PingAccess. PAPM requires at least one environment.

Steps

1. Using the Settings tool, click **Environment Settings** on the left navigation menu. The environments previously created will display as buttons on the screen.
2. Click **Create New**.
3. Input the information for the new environment and click **Save**.
   The environment builds. Once complete, a success message displays.

The system will display a message if it is unable to connect to an environment. If you are unable to connect to an environment, check the following:

- Ensure the hostname is correct and that it is reachable.

- Ensure the hostname matches that of the TLS certificate. The certificate hostname should match the hostname for the system on which the product is installed.
- Ensure the ports are correct and accessible.
- Ensure usernames and passwords are correct.

License management

Use the **License Management** screen to add or update a new PAPM license file.

See *Import license* on page 395 for more information.

Application settings

The **Application Settings** menu configures PAPM authentication. To access this screen, navigate to **Settings# Application Settings**.

- Select from the **Policy Suite Sign-on** list to specify the active login module.
- To change modules, select the module and click **Save Configuration**.

Note: PAPM supports user accounts and password storage in either an embedded database or via LDAP.

Database Login

Database Login is the default login type when PAPM installs for the first time. We recommend using LDAP as the user store for enhanced security and availability.

- Administrators give access to users by assigning username and password combinations for each user in the user interface.

LDAP Login

LDAP Login module provides simple authentication with group-based authorization. Configure the following values to enable LDAP Login. More information on these values can be found in Spring Security documentation found *here* and *here*.

Note: In the event that LDAP Login is misconfigured, resulting in a lockout, you can force Database Login by setting `papm.force-database-authentication` to `true` in `application.properties`.

Field	Description
LDAP URL	URL of the LDAP Server. For example: `ldap://example.com:389/dc=example,dc=com`.
USER DN PATTERN	The login user's DN pattern. For example: `uid={0},ou=people`. `{0}` must be present and will be substituted with the username. Optional if User Search Filter is configured.
USER SEARCH FILTER	The login users search filter. For example: `(uid={0})`. `{0}` will be substituted with the username. Optional if User DN Pattern is configured.
USER SEARCH BASE	The search base for user searches. Only used with a User Search Filter. If omitted, defaults to root search.

Field	Description
GROUP SEARCH FILTER	The group search filter. For example: `(uniqueMember={0})`. `{0}` will be substituted with the DN of the user. `{1}` can be used if you want to filter on the login name.
GROUP SEARCH BASE	The search base for group membership searches. If omitted, defaults to root search.
GROUP ATTRIBUTE	The LDAP attribute name which contains the group name. For example: `cn`
ALLOWED GROUPS	The groups allowed for access to PAPM. Separate multiple groups with '|'. Allow any group with '*'.
MANAGER DN	The DN of the LDAP account used by PAPM to connect to the LDAP server. Optional if anonymous access is used.
MANAGER PASSWORD	The password for the LDAP account. Required only if Manager DN is specified.

Change login type

About this task

When PAPM is first installed, the login type is set to Database Login. However, **LDAP Login** is the most common login type for PAPM. This section describes the process to change the login type to LDAP Login.

Steps

1. Using Settings, click the **Application Settings** link from the left navigation menu.
   The **Application Settings** screen displays.
2. Select the desired login type in **Policy Suite Sign-on** list box.
   The information needed for the selected login type displays. See above for detailed login type information.
3. Type login type information and click **Save Configuration**.

   Note: You must input valid LDAP information to prevent a lockout when changing login type. It is possible to leave a browser logged in and open a second browser to test the login type change.

   Note: In the event that LDAP Login is misconfigured, resulting in a lockout, you can force Database Login by setting `papm.force-database-authentication` to `true` in `application.properties`, and restarting PAPM.
4. If Secure LDAP is used, add the trusted certificate used by the ldap service to the JRE truststore:

   For example:

   ```
   keytool -import -trustcacerts -file /path/to/ca/ca.pem -alias CA_ALIAS -
   keystore $JAVA_HOME/jre/lib/security/cacerts
   ```

Database management

Database Management contains the following functionality:

- **Current Statistics:** View the database estimated row count
- **Archive Settings:** Manage the size of PAPM databases.

Settings, audit records, and application data are stored in the database. The database auto-archives itself every day and maintains 10 backup copies in the `<PAPM_HOME>/config/DB/BACKUP` directory. Manual user intervention is not required for this backup to occur.

Archive Settings

To manage the growth of the database, use Archive Settings. For each of the tools that use the database, input a maximum number of records (rows). PAPM checks the row count nightly to determine if the row count exceeds the limit set by the user. If it does, PAPM creates a CSV archive file containing the oldest inserted records in the database. The number of records in the archive file is the number of records that are over the limit.

For example, two records are added to a database containing six records. If the user has set the record limit to six, the additional two records exceed the record limit. PAPM archives this data at the end of the day by creating a CSV file containing the data for the first two records (the oldest records) in the database. Next, PAPM removes these records from the database and brings the record count back down to the limit set by the user.

The size of an audit record can be between 500 bytes and 20,000 bytes. An archival setting of 10,000 records with an average size of 10,000 bytes per record will result in 100MB of data.

Manage the size of the database

About this task

To manage the size of the database that stores PAPM data, set the maximum number of records for each database.

Steps

1. Using the Settings, click the **Database Management** link from the left navigation menu.
2. Click **Archival Settings**.
3. For each of the tools, type a value that the tool must reach before it starts archiving the data for that tool.

Customize the number of database backups

About this task
To customize the number of database backups:

Steps

1. Open `<PAPM_HOME>/config/application.properties`
2. Using the `papm.database.max-backups` property, specify the maximum number of backups.

Revert to a previous backup

About this task

To revert to a previous backup:

Steps

1. Shut down PAPM.
2. Remove or backup the associated database files (all end in *.db, *.lock, *.part), in the following directory: `<PAPM_HOME>/data`
3. Copy the backup ZIP file from the `<PAPM_HOME>/data/backup` directory into `<PAPM_HOME>/data`.
4. Unzip the file into `<PAPM_HOME>/data`

5. Restart PAPM.

Change configuration database passwords

About this task

The PingAccess Policy Migration configuration database is protected by two passwords - a file password and a user password. These passwords both default to `2Federate`, but should be changed for production environments. To keep your data secure, change the password to the PAPM database.

Note: Changing either password requires PingAccess Policy Migration to be shut down.

Tip: It is recommended to use a key pair with a different password from the system default, and further recommended to obfuscate this password using the `obfuscate.bat`/`obfuscate.sh` utility in `<PAPM_HOME>/bin`. Add the obfuscated password to `<PAPM_HOME>/config/application.properties`.

Steps

1. Open a terminal window and change to the `<PAPM_HOME>/bin` directory.
2. Ensure that the `JAVA_HOME` environment variable is set correctly by executing the command `echo $JAVA_HOME`.
3. Ensure that the proper Oracle Java executable is in your path. Enter the command `java -version`. If this command returns a value indicating that the Java executable is not a supported version of Oracle Java, correct this issue before continuing.
4. Shut down PingAccess Policy Migration.
5. Optional: To change the database file password, use the following commands:
   - On Windows: `dbfilepasswd.bat old_password new_password`
   - On Linux: `./dbfilepasswd.sh old_password new_password`
6. Optional: If you changed the database file password, update the `papm.database.file-password` property in `<PAPM_HOME>/config/application.properties` with the obfuscated password output from the command used in the preceding step.
7. Optional: To change the database user password, use the following commands:
   - On Windows: `dbuserpasswd.bat file_password old_password new_password`
   - On Linux: `./dbuserpasswd.sh file_password old_password new_password`
8. Optional: If you changed the database user password, update the `papm.database.password` property in `<PAPM_HOME>/config/application.properties` with the obfuscated password output from the command used in the preceding step.

SSL certificates

Steps

1. Create a self-signed certificate using the Java Keytool. The following command generates a key pair and certificate directly into file keystore. For example:

```
keytool -keystore config/keystore.p12 -alias papm -genkey -keyalg RSA
```

Note: If a file named keystore already exists, make a backup. This command prompts you for information about the certificate and for passwords to protect both the keystore and the keys within it. The only mandatory response is to provide the fully qualified host name of the server at the "first and last name" prompt.

2. Generate a CSR (Certificate Signing Request). For example:

```
keytool -certreq -alias papm -keystore config/keystore.p12 -file papm.csr
```

> **Note:** The default location of the keystore is `<PAPM_HOME>/config`.

3. Send the CSR to a certificate authority to be signed.
4. Obtain the signed certificate.
5. After receiving the signed certificate, load it into the truststore using the following command:

```
keytool -keystore config/keystore.p12 -import -alias papm -file papm.crt -
trustcacerts
```

6. To obfuscate the password, navigate to the `<PAPM_HOME>/bin` directory, type the applicable filename and press **Enter**. You will be prompted to enter the password you want to obfuscate.
   - **Windows**: `obfuscate.bat`
   - **Linux**: obfuscate.sh
7. Open *PAPM_HOME*`/config/application.properties`and configure or update the SSL properties.
   - server.ssl.key-store=${papm.home}/config/keystore.p12
   - server.ssl.key-store-type=PKCS12
   - server.ssl.key-store-password=2Federate
   - server.ssl.key-password=2Federate
   - server.ssl.key-alias=papm
8. Start the server by executing the following command:

```
./run.sh
```

Version

This section displays PAPM version information.

Monitoring Dashboard

The PAPM Monitoring Dashboard provides real-time environment status for Ping Identity services. It also provides a historical view of activity to aid in troubleshooting. The monitoring console is designed to be flexible enough to work with third-party web applications.

Summary health check

Summary health check displays the status of active monitors by their environment and status. It provides a single health status view of the various Ping servers. This view can quickly alert an administrator to current or escalating issues.

Health check status indicators

A status color of Green, Yellow, or Red is assigned to each service URL identified by PAPM in the Active Monitor list.

Healthy (Green)	The Green status indicator denotes a healthy active monitor. Monitoring Dashboard has not identified any issues with connectivity, use, etc.

Sync Failure (Yellow)	The Yellow status indicator denotes a sync failure. In the event of a sync failure, the monitor will set the entire environment to this status. Check the Sync Status UI for details on the failed component(s). If an environment has both a sync failure AND a status failure, then the status of the environment will change from yellow to red.
Active Failure (Red)	In the event of an active failure, the monitor will set the environment to red and move the offending service check to the top of the Active Monitor queue. The dashboard only shows heartbeat data for an environment by default. If an offending monitor does not show up on the list of active monitors, remove the heartbeat search filter and refresh.

Sort Order

The Active Monitor list displays an inventory of all known services. Issues across the system or within a given environment display at the top of the list. This list sorts in the following order:

1. **Status**: If any individual monitor has an error response, it sorts first.
2. **Environment**: All monitors sort by their environment definition next.
3. **Response time:** Finally, the list sorts by the response time. Hence, the first successful entry for any environment will be the slowest responding service.

To configure a summary health check, see *Add a new monitor* on page 407.

ⓘ **Tip:** Define flexible environments for improved readability. For example, instead of defining an environment like 'prod', define multiple environments like 'prod-dc1' and 'prod-dc2' to isolate issues within a specific data center and enhance native sorting based on response times.

Response times

Response Times provides a graphical display of historical and live response times as well as CPU Load and Open Connections comparisons for Ping servers. Monitor URLs that connect to a PingAccess proxy port heartbeat to see the activity of open proxy connections.

To view response time data, you must have active monitors in an environment.

1. Select the **Environment** for which you want to view response time data.
2. Select the monitor **Name**(s).
3. Specify whether monitoring is **Enabled** or **Disabled**.
4. Specify the **URLs** you want to monitor.
5. Click **Draw Charts**.
6. Review historical data by specifying a date range in the dropdown.

To configure a response time display, see *Add a new monitor* on page 407

Memory usage

Memory usage charts show the total, free, and used system memory available to the JVM. A healthy JVM should show regular garbage collections and variable memory usage related to system utilization, but not a net increase of memory usage over time.

The system monitor provides an overview of the available memory and its utilization. If JVM memory does not show a healthy garbage collection process or free memory, ensure the JVM argument sizing is appropriate in the Ping start scripts.

To view memory usage data, you must have active monitors in an environment.

1. Select the **Environment** for which you want to view response time data.
2. Select the monitor **Name**(s).
3. Specify whether monitoring is **Enabled** or **Disabled**.
4. Specify the **URLs** you want to monitor.
5. Click **Draw Charts**.
6. Review historical data by specifying a date range in the dropdown.

To create a memory usage monitor, see *Add a new monitor* on page 407

Sync status

Sync status offers a Ping-specific monitoring feature that ensures servers within an environment (cluster) are synchronizing PingFederate and PingAccess JSON web keys and PingAccess application policies appropriately. It automatically places heartbeat endpoints based on the configured monitor URL and the assigned environment. The first server in the sync group determines the reference version of data.

- **Run Comparison:** Click to refresh the screen to get a real-time current view of the status.
- **PingAccess Database Sync:** Validates the heartbeat endpoints for an environment to ensure that they contain the same number of configured applications, virtual hosts, and proxy sites.
- **PingAccess JWKS Sync:** Tests all JWKS endpoints within an environment to ensure that all services are using the same set of keys.
- **PingFederate JWKS Sync:** Tests all JWKS endpoints within an environment to ensure that all services are using the same set of keys.

To create a sync status monitor, see *Add a new monitor* on page 407

Note: If servers that do not belong in the same cluster are paired under the same monitoring environment, this console will always show failures.

Note: If a reference server is the endpoint that is out of sync, the UI will mistakenly show all of the other servers as out of sync. It is not possible to query the admin node via a web interface for the correct reference data.

Search

Search provides an interface to search historical events and responses. You can search and sort archived data using predefined parameters.

- Select a **Monitor** and click the arrow to render available dates.
- Specify a date to search.
- Click **Display Data**.
- In the search results list, sort by event times, response times, and status
- Click the **Event Time** hyperlink to inspect the text response from any monitor.

Manage monitors

Add a new monitor from **Manage Monitors**. Three MB of disc space of data per monitor per day on a 60-second interval is required.

Custom monitoring URLs have different storage requirements related to the page size. The monitor console has a download limit of 10 KB per page. If the test URL needs to download more than 10 KB of data, then it will fail. This limitation prevents storage exhaustion.

- **Environment:** Each monitor URL should be tied to the correct environment (e.g. Dev, Test, QA, Prod). The environment is more than a descriptor; it is also used to sort errors and eventually feed the health status of the entire environment.
- **Monitor Name**: The Monitor Name should include a descriptive title that is searchable. For example: "`Svc Name` *`ServerName`*" "`PA JWKS Server15`"
- **Monitoring URL:** The Monitoring URL is the endpoint that the monitor will call to test the target service.
- URLs that contain the strings: "`/pf/JWKS, /pa/JWKS, /pa/heartbeat.ping`" are treated differently from other URLs and are evaluated in the key and engine sync tests.
- **Monitoring Interval**: Monitoring Interval is used to configure the wait time between tests. This value directly affects the data storage requirements. For example, a 30 second interval will require 2x as much storage as a 60 second interval.
- **Success Criteria**: A monitor reports success or failure based on this attribute.
  - `Default`: HTTP/200 Response Code (will succeed with any response code in the 200 range)
  - `Contains`: Must contain the case-sensitive, substring in the HTTP Response Body (not in the response headers).
  - `Regex`: Java Regex match within the HTTP Response Body. Use the test Monitor function to refine the regex. Example: `.*("number.of.virtual.hosts": "[0-9][0-9]").`
- **Enabled**: A monitor should generally always be enabled. Use this option to disable a monitor during known system downtimes to suppress errors and alerts.
- **Username / Password:** Add a username and password if the resource requires the basic authentication header to be added to the request. Passwords are encrypted in the database.
- **Additional Header**: Some service calls may require additional headers. For example, an API test against the PingAccess APIs requires a X-Xsrf header.
- **Proxy & Archival Settings** button: See *Proxy and archive settings* on page 407.

PingAccess enhanced heartbeat messages setup

About this task

Steps

1. In PingAccess, open `<PA_ROOT>/conf/run.properties` and set `detailed.heartbeat.response` to **true** .
2. Configure `<PA_ROOT>/conf/template/heartbeat.page.json` to have the appropriate attributes.
3. Save the files and restart PingAccess.

> **Tip:** Although hostname is not required, it is best practice to add a unique, opaque identifier in the hostname field for agent/proxy identification.

PingFederate enhanced heartbeat messages setup

About this task

Steps

1. On the PingFederate server, open `<PF_ROOT>/bin/run.properties` and set `heartbeat.system.monitoring` to **true.**
2. Ensure the heartbeat template includes at minimum the parameters to measure memory and load statistics. Open the `<PF_ROOT>/server/default/conf/template/heartbeat.page.template` and check that the memory and load parameters are available in the template.
3. Additional parameters can be configured, but may not be used directly by the Monitoring Dashboard.

4. Save the files and restart PingFederate.

> **Tip:** If your site has already configured load balancers to use the “OK” response message from the default heartbeat template, then simply add the “OK” text into the template.

Add a new monitor

About this task

1. Using Monitoring Dashboard, click the **Manage Monitors** link from the left navigation menu.
2. Input information for the new monitor in the **Create New Monitor** screen. See *Manage monitors* on page 405 for more information.
3. Click **Test Monitor** to ensure the monitor works as expected.
4. Click the **Create New Monitor**

Delete a monitor

About this task

Steps

1. Using Monitoring Dashboard, click the **Manage Monitors** link from the left navigation menu.
2. In the **Existing Monitors** list, click the delete icon for the monitor to remove.

Proxy and archive settings

The monitoring service can be configured to use a forward proxy to test URLs.

- If a proxy is set, monitors will attempt to use the proxy interface.
- Configures the number of days to retain historical data.
- This is a global setting and affects all monitors.

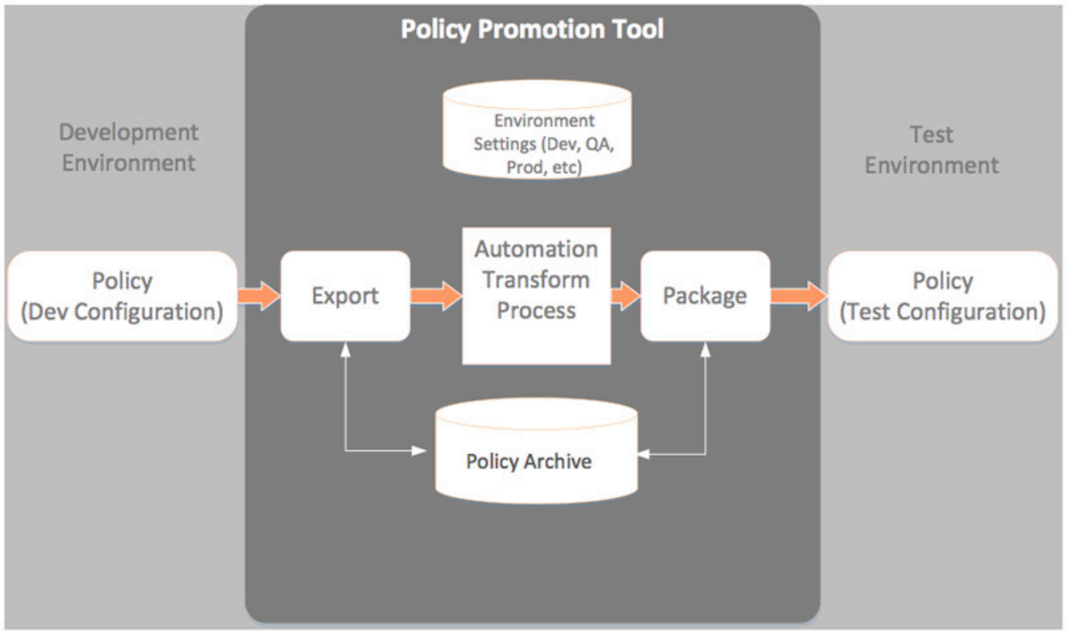
To access these settings, navigate to **Monitoring Dashboard**# **Manage Monitors** and click the **Proxy & Archive Settings** button.

Policy Automation

Policy automation allows the administrator to migrate configuration elements for PingAccess, to migrate all of the configuration elements for a PingAccess application based on templates, set conditions on what to do in target environments when changing PingFederate/PingAccess configuration, and to promote policy between operational environments. It can also export policies and store them in an auditable flat file database or show the difference between two policies.

Policy promotion process overview

Policy migration is a web based deployment and management solution that works with PingFederate and PingAccess. This diagram shows the typical process flow for policy promotion from DEV to TEST environments.


Policy Promotion Tool
Development Environment
Environment Settings (Dev, QA, Prod, etc)
Test Environment
Policy (Dev Configuration)
Export
Automation Transform Process
Package
Policy (Test Configuration)
Policy Archive

An application policy and all of its dependent policy objects are exported from the development environment into the PAPM database. The PAPM module transforms the source environment data to match the target TEST environment and calls the APIs to import the new application policy objects. The application and all of its required components are now available in the target environment.

Tip: PingAccess Policy Migration supports dynamic policies to handle conflict conditions. For example, during the import process, if there is already an existing object, PAPM can be configured to link to the object instead of overwriting it. In the case of a failure condition, PAPM can either roll an object back or leave the updated data in place.

Note: PAPM uses the management APIs of PingFederate and PingAccess. Make sure the required administrative consoles are running and accessible from the PAPM.

To ensure accuracy, validate the following settings after a policy is promoted using PAPM:

- Virtual host Certificates
- Site Certificates
- Site Load Balancing
- Websession Cookie Domain

Policy Operation

Promote, export, and remove applications from the **Policy Automation**# **Policy Operation** screen.

The first section of the **Policy Operation** screen defines the source application:

1. **Environment:** Displays list of available environments. Select the environment containing the applications that need to be loaded.
2. **Connection Type**: Select the application type. Available applications for the selected environment and connection type populate the **Connection** list box.
3. **Connection:** Select the application policy from the list of available applications from this environment
    - **View Application Details:** Click this icon to display application artifacts in a pop-up window.

The next two sections of the **Policy Operation** screen define objects selected for the application in the first section.

1. **List Releases** button: A release occurs when an export promotes to an environment. Click to load the list of packages released for the selected application.
2. **List Exports** button: A exports is a version the application at a point in time. Click to display all prior exports of this application.

> **Note:** You must create at least one export in order to populate this list.

3. **Select Package**: Select the package to release into target environment.
   - View Package Details icon: Click to display details of export/package in a pop-up window.
4. **Target Environment:**
   - **Change Number:** Type the release label for the target environment.
   - **Select a target environment**
   - **Publish:** Click this button to push the selected export to target environment.
5. **Upload Package** button: Click to select a package to upload.

Action buttons are found at the bottom of the screen.

1. **Export Application**: Click this button to export current application from this environment and creates an export with unique identifier as per below format: `[YYYY-MM-DD HH:MM:SS.zzz>sequencing number]. g.:2017-02-01 21:30:56.233 > 238`
2. **Remove Application:** Click this button to remove the selected application from source environment. It removes all dependent components as well.

Publish Package

The **Publish Package** screen displays when clicking the **Publish** button from the **Policy Operation** screen to allow users to configure the publish operation. For example, users can change the target application name if needed.

Load an application

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu.
2. From the **Environment** drop-down list box, select an environment.
3. Click the **Environment load** button to load the selected environment.
   The applications for the selected environment populate the applications drop-down list box.
4. From **Connection Type** drop-down list box, select connection type.
5. From the **Connections** drop-down list box, select an application.

Promote a policy

About this task
The Policy Promotion component automates policy promotion from one environment to another. This component automatically assigns environment specific virtual host names and a target proxy site. Use this procedure to promote a new policy or publish a previous version of a policy.

Steps

1. Click the **Policy Operation** link from the left navigation menu and follow the steps to load the application.
2. Click the **List Exports** button to load the available exports for this application.

   The available exports populate the page. If there are no previous exports at this point, create the export first.

3. Select the export from the list of available exports of the **Package Selection**.
   The export details populate the page.
4. (Optional) Type a release number in the **Target Environment Change Number** text box. Releases should be unique and easily identifiable.
5. Select the **Target**. If promoting a policy to a new environment, the target environment will differ from the source environment selected when loading the application in the first step.
6. Click the **Publish** button below the Target Environment list box.
   The **Publish Package** screen displays.
7. On the **Publish Package** screen, choose **Active** or **Inactive**. If needed, modify the **Application Name** and **Context**.
8. Select an **Operational Policy** from the **Policy** list box. Operational Policies are user created in PAPM and define how PAPM handles object during processing.
9. Type a **Virtual Host** and **Site Target**(s).
10. Click the **Publish** button.
11. Validate the application published to the target environment.

    - **Target Application Name:** Populates with the source application name and should not typically be changed. Users can customize the target application name to include the name of the target environment. For example, change the name from AppDev1 to AppDev2. This can be useful to support multiple application environments with a single Ping environment.
    - **Policy:** Operational policies specify how PAPM will behave during create and push operations. See *Building operational policy* on page 414 for more information.

If the policy fails to publish, details of the failure are documented in **Audit Data**# **Other Audit**. Pick the link associated with the application name that you were attempting to publish. Details of the configuration are documented on the PingAccess or PingFederate Release Data pages.

View application details

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu and follow the steps to load the application.
2. Click the **View Details** button located on the right side of the **Connections** dropdown list.

Create an export

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu and follow the steps to load the application.
2. Click the **Export Application** button to export the application from source environment.

   For IDP and SP connections, the policy is exported.

   If the selected application is a PingAccess API, The **Select OIDC Client for API application** screen displays.
3. For PingAccess applications, select an OIDC client from the list then click **Export Application** to export the policy.

View the export list

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu and follow the steps to load the application.

2. To load the export list, click **List Exports**.
   Previous exports populate the **Exports**# **ConnectionType**# **Env** list.

> **Note:** A warning is displayed if no exports exist. To create an export, see *Create an export* on page 410.

3. Click the **Exports**# **ConnectionType**# **Env** Down Arrow to view the list.

Download a package

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu and follow the steps to load the application.
2. To load the export (package) list, click **List Exports**.
   Previous exports populate the **Exports**# **ConnectionType**# **Env** list.

> **Note:** A warning is displayed if no exports exist. To create an export, see *Create an export* on page 410.

3. Select an export from the **Exports ConnectionType Env** list.
4. Click the **Download** icon for the selected export.
   The export downloads to the downloads folder.

Upload a package

Steps

1. Using Policy Automation, click the **Policy Operation** link from the left navigation menu.
2. Click **Upload Package**.
   The **Publish Package** screen displays.
3. Click **Publish**.

Policy Settings

Policy migration requires a connection to PingFederate and PingAccess environments to access API data. These connections allow PAPM to export policy data, manage policies, and push policies to target environments such as DEV or TEST. See *Settings* on page 396 for more information.

The procedures in this section are used to configure components that are required for policy creation.

vHost management

To enable 1-Click operations, PAPM tracks changing environment data in its own database. vHost management updates virtual host data and synchronize that data to both PingAccess (virtual host entries) and PingFederate (whitelisted URLs in the OIDC client).

To access this screen, navigate to **Policy Automation**# **System Settings**# **Virtual Host Management (PingAccess)**.

- **Environment**: Contains available environments.
- **Select Application**: Contains available applications for the selected environment
- **Edit Virtual Host**: Enter the modifications for host name or port number as host:port
- **Virtual Host: Policy Tool**: This displays existing virtual hosts configured for the selected application.
- **Update VHost**: Updates any necessary changes to existing virtual host configurations in PingFederate and PingAccess.
- **Remove VHost**: Enter the virtual host and port pair then click the **Remove vHost** button to delete the vHost. You cannot delete a virtual host that is assigned to an application.

Manage vHosts

Steps

1. Using the Policy Automation, click the **System Settings** link then select **vHost Management** from the left navigation menu.
   The **Virtual Host Management (PingAccess)** screen displays.
2. Select an environment from the **Environment** drop-down list box.
   The applications for the selected environment populate the **Select Applications** drop-down list box.
3. From the **Select Applications** drop-down list box, select an application.
   Name and port number display for the selected application.
4. To add a virtual host, click the **Add** link, specify the `virtual host:port` combination, and click **Update**.
5. To update the host name or the port number, enter the necessary changes to host name or port number in the editable text box, and click the **Update VHost**.
   The host name, port number, or both are updated.
6. To delete a virtual host, click the **Delete** icon to the right of the virtual host entry in the list.

> **Note:** Each application requires at least one virtual host setup. Default virtual host configuration cannot be deleted.

Site management

The **Site Management** screen updates proxy targets and synchronizes that data to both PingAccess and PingFederate.

- **Environment**: This drop down list display list of all environments that were configured previously on setup.
- **Select Application**: This drop down list of available applications from the selected environment.
- **Edit Site:** Enter the modifications for host name and port number pair as **host:port**
- **Add Site**: Click the **Add** link and enter the `host:port` combination for the site.
- **Update Targets:** Click the Update Targets button to update to target environment.
- **Update Site Configurations:** Click this button to display complete site configuration for this application in a new screen and that allows modifications to site configuration.

Site Update

The Site Update screen provides a way for users to input a new site. You can also update a site or delete a site that is not in use by an application. To access this screen, click **Update Configurations**.

- **Targets**: Multiple targets separated by a comma.

Certificate management

PingFederate makes extensive use of digital certificates. Configure PAPM with default signing and decryption certificates used for 1-Click creation of SAML connections. The **Certificate Management** screens provide an interface to manage these system defaults.

- **Signing and Decryption Keys and Certificates:** Contains the list of certificates available for signing and decryption. In PingFederate, these establish and maintain the server's signing certificates, which may be used to sign assertions, security tokens, requests, and responses. These certificates may also be used for decryption.
- **SSL Client Keys & Certificates**: Contains a list of client keys and certificates used for TLS authentication to systems that may require it.
- **Import Certificate (PKCS12)**: Import a certificate.

Add a new certificate

Steps

1. Using Policy Automation, click the **System Settings** link to open this section in the left navigation menu then click **Certificate Management**.
2. Select an **Environment**.
3. Click **Import Certificate**.
4. Select a certificate type then click the **Select Certificate File** button to select a certificate file.
5. If needed, enter a password.
6. Click the **Save Certificate**.

Datastore management

The **Datastore Management** screen maps unique LDAP and database IDs from the various environments. As PingFederate SAML connections or OAuth configurations are pushed from one environment (DEV) to another (TEST), the system needs to dynamically map these unique IDs. This function is critical to the 1-Click Promotion capability.

Note: Datastore mappings should be setup with the initial system deployment and updated whenever new datastores are added.

- **Map Datastore:** Select a source and target environment to map. The UI reads the available data store IDs and makes them available for simplified mapping.

Note: Forward and reverse mappings are required. For example, in a deployment with two environments (DEV, PROD), you must map the data stores from DEV to PROD. If you ever plan to push a policy from PROD back to DEV, then you must map the data stores in the opposite direction as well (PROD to DEV).

Map environments

About this task
The following process describes how data stores map between environments.

Steps

1. Using Policy Automation, click **System Settings**, then click the **Datastore Management** link from the left navigation menu.
2. Select an Environment (Source) and Environment (Target) and click **Map Datastore**
3. For both environments, select a JDBC and LDAP.
4. Click **Save Configuration**.

Template management

Templates are the basis of all single click application creations. The vast majority of application policies share a number of attributes. Using well-formed templates that match the security rules and internal security policies can revolutionize the operational workflow for policy administrators.

To access this screen, navigate to **Policy Automation**# **System Settings**# **Template Management**.

Create Template Application

The Create Template Application function will automatically generate a PingAccess application policy and all of its dependent components in a well-named, portable format. You can modify this policy to meet requirements. This quick start method prevents administrators from making small errors in naming. Once the application is created, you can manage it from PingAccess.

Add Application as Template

This function marks an existing application policy as a template and makes it available as a template in the **Migration** UI.

- Click the **Add Application** icon to associate the selected application with the template.

Delete Application as Template

This function removes an existing template. The application that the template is based on is not deleted.

- Click the **Delete icon** to delete the selected template (does not delete the application policy).

Create a template

About this task
PAPM allows developers to define templates for applications. This can be very useful to onboard similar applications. Existing applications can be marked as a template and applications are created using the existing applications settings.

PAPM templates are based on a three-step principle:

- **Step 1: Create Template**: Create the template from existing application.
- **Step 2: Create Application from Template**: Create the new applications from template. This eliminates the manual steps involved in creating OIDC configuration, web session configuration, and identity mappings configurations for applications of similar setups.
- **Step 3: Modify New Application:** Update and fine-tune the application created.

Steps

1. To create a template: Using Policy Automation, click the **System Settings** link then select **Template Management** from the left navigation menu.
2. From the **Environment** drop-down list box, select an environment. Click the **Environment load** button to load the available applications.
3. Click **Create Template Application** to expand the template create section.
4. For PingAccess applications, select the type of template. **Agent** or **Site**. The **Create Template Application** function is only available for PingAccess as it helps to generate a number of complex policy objects.
5. Type the name of the template application in the text box.
6. Click the **Create** button to create the template application. At this point **Template Application** does not have functioning application data associated. You may need to update the host name or virtual host. The following message displays: **Creating Template Application successful**.
7. Click **Add Application AS Template** to expand **Add Template**. Select the Application.
8. Click the button to associate specified application data with the template application.
9. Use the PingAccess and PingFederate Administration Consoles to modify the template configurations as needed.

Building operational policy

As policy data is pushed around between environments, Policy Automation must deal with conflicts. For example, if a policy is promoted and one of the API calls fails, should PAPM roll the policy back to the original data or leave the partial data in place? If existing objects are found, are they overwritten or reused? Operational Policies manage these situations.

To access this screen, navigate to **Policy Automation**# **System Settings**# **Build Operational Policy**.

- **Policy Type**: Choose the policy type as **PingFederate, PingAccess,** or **OIDC**.
- **Operation**: Select **Create-Operation** or **Push-Operation** for the policy operation
- **Policy Name**: Contains the name of the operational policy.

- The list in the **Create/Edit Policies** section and the list of saved actions in the **Existing Policies** section change based on the policy type and operation selected.
- To create a new operational policy, enter a policy name, and choose settings in the Create/Edit Policies section and click the **Save Policy** button.
- **Create-Operation Actions:**
  - **Delete on failure:** Applies to Rollback. If the policy build fails when PAPM is creating a Ping policy set, select this action to delete the object set.
  - **Keep Objects:** Applies to Rollback. If the policy build fails when PAPM is creating a Ping policy set, select this action to keep the objects instead or deleting them.
  - **Re-Use if present:** Applies to all actions except for Rollback**.** When PAPM is creating a Ping policy set, PAPM will re-use the objects found in the template.
  - **Create New:** Applies to all actions except for Rollback**.** When PAPM is creating a Ping policy set, if an object with the same name already exists, create a new object instead re-using the existing object. This new object is assigned a unique name based on the input application name.
- **Push-Operation Actions:**
  - **Revert to Original:** Applies to Rollback. If an error occurs that prevents PAPM from completing the push of a policy to a new environment, remove any objects that created / updated and keep existing objects.
  - **No Change:** Applies to Rollback. If an error occurs that prevents PAPM from completing the push of a policy to a new environment, save the partially updated objects in place.
  - **Link if Present:** Applies to all actions except for Rollback**.** When pushing a policy to a new environment, link to existing objects instead of updating or overwriting objects.
  - **Update if Present:** Applies to all actions except for Rollback**.** When pushing an operation to a new environment, replace existing objects with the new objects included in the push.

Building an operational policy

Most create-operation type operational policies should create new objects and most push-operation type polices should update objects in place rather than linking to existing objects. While this will result in more objects and a larger policy store, the store will be more portable, stable, and predictable. Imagine pushing a policy that results in no behavior change or that results in multiple policies changing because they share a dependency.

1. Using Policy Automation, click the **System Settings** link then select **Build Operational Policy** from the left navigation menu.

   The **Build Operational Policy** screen displays.
2. Select a policy type.
3. Select the policy action.

   The fields in the **Create/Edit Policies** section change based on these selections.
4. Enter a name for the operational policy in the **Policy Name** text box.
5. Select an action for the remainder of the fields in the **Create/Edit Policies**
6. Click the **Save Policy** button.

Compare Exports

About this task

Compare Exports finds differences in application policy versions or SAML connections**.** For example, it will show the differences between an application in the development and production environments.

- **Environment**: This drop down list displays list of all environments previously configured.
- **Application Name**: This drop down list displays all applications for selected environment.
- **Export:** This drop down list displays available exports of selected application.
- **Compare:** Click this button to compare the selected exports for differences.

- **Results:** This section is the third column on the screen and it displays the differences (highlighted).

> **Note: Cipher Data:** As policies are pushed and shared, secrets or keys dynamically generate and change. This cipher data can show up as a false positive and showing differences in the compare. Ignore cipher data.

Steps

1. Using Policy Automation, click the **Compare Exports** link from the left navigation menu.
   The **Compare Exports** screen displays.
2. Select a connection type: **Ping-Access, IDP,** or **SP**
3. From the **Environment** drop-down list box, select an environment.
4. From the **Select Applications** drop-down list box, select an application.
   Available exports for the selected application populate the **Select Export** list box.
5. Select an export from the **Select Export** drop-down list box.
6. Repeat steps 2 through 5 to setup the second part of the compare.
7. Click the **Compare**.
   The differences between two exports display (highlighted) in the third column of **Compare Results**.

Audit Data

Audit Data is broken down into three parts: PingAccess Release Data, PingFederate Release Data, and Other Data. The **Release Data** sections display the records of every application policy or SAML connection that was created or promoted. The **Other Audit** data contains every other type of call to PingFederate and PingAccess.

> **Note:** The **Other Audit** section is indispensable for troubleshooting application failures. Without checking log files, this UI can show the root cause of most Policy Automation failure.

- **Audit Data:** Click this menu item on left menu to display Audit Data screen. Click an application name for more details.
- **Release Data:** Click to view PingAccess or PingFederate log details for releasing packages to environments. Click an application name for more details.
- **Other Audit:** Provides all other log messages.
- **Search Filter:** Enter the search term to narrow down list of system messages. Example: enter **DEV** to show messages that include the string
- **Application Name hyperlink**: Click an **Application Name** link to inspect application data in a pop-up screen.

Policy Testing

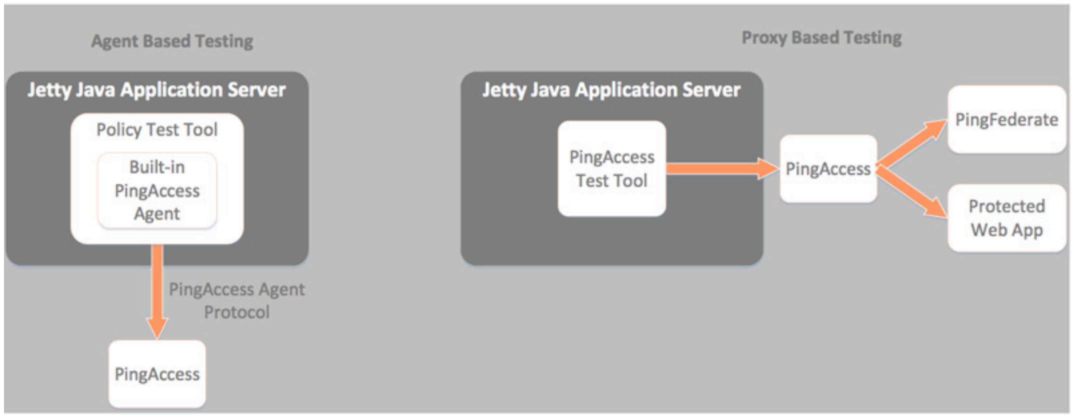
The PAPM Policy Test Tool (PTT) tests application policies in a given environment. The response is returned with any SSO Headers that might be configured for the policy. The Policy Test Tool enables agile policy development since policy logic can be vetted in advance of agent deployments. The Load Test suite provides administrators the insight to understand a performance impact of a given policy.

> **Tip:** Proxy Based applications must be physically deployed to the network. The PingAccess engine does not allow proxy resources to be tested in the same manner as agent deployments.

Architecture overview

The PTT has a built-in agent that leverages the PingAccess Agent Protocol. It is able to instantiate itself and act like any other agent on the network. For Proxy-based applications, the PTT sends requests over the proxy connection. Due to the difference in architecture, the protected application must be running and available behind the PingAccess Proxy.

The following diagram shows the differences between the Agent-based and Proxy-based testing:



The PTT has a built-in PingAccess Agent that talks to PingAccess Policy Server over PingAccess Agent Protocol (PAAP) to evaluate the Policies. PAAP is a simple HTTP based protocol for communication and interaction between PingAccess Policy Server and PingAccess agents. The Agent Configuration is downloaded and stored in a file, which is used by the built-in Agent to communicate with PingAccess for Policy evaluation.

A protected web application is required to test proxy resources. PTT communicates with PingAccess using HTTP protocol and evaluates the policies.

See *Agent Vs Proxy Testing* in the troubleshooting section for more information.

Settings

Policy Test Tool (PTT) identifies how a user authenticates by selecting adapters or authentication context it needs to pass with the request to determine the adapter for user authentication. Customize authentication adapters for PTT from the **Policy Test Tool# Settings** screen.

- **Environment:** Choose an environment for the adapter configuration. Each environment is setup individually.
- **Adapter:** Configures the values that the PTT uses to auto-fill the authentication.
  - Allows the PTT to abstract the entire authentication process from the user and show the complete UI on a single screen and without redirections.
  - For the default authentication adapter, select an IdP Adapter Mapping created in PingFederate for user Authentication.
  - If the authentication context ACR is not defined, PingAccess Policy Tool uses the Default Authentication Adapter defined
- **Configuration**: Defines how the credentials pass.
  - HTML configuration is recommended since Basic Configuration is used where the user is challenged for credentials with basic pop-up
  - Uses the basic configuration setting if the authentication mechanism uses the authorization header.

Create adapters

About this task
An Authentication Context (ACR), a Default Authentication Adapter, and a PingFederate Adapter are input from **Settings**.

Steps

1. Using the Policy Test Tool, click **Settings** from the left navigation menu.
2. Select an authentication context or adapter to open the section.
    - **Authentication Context (ACR):** Authentication Context (ACR) settings determine the adapter that PingFederate will use for Authentication while evaluating a policy. If the ACR is not setup for an application, the default Authentication Adapter below is used.
    - **Default Authentication Adapter**: If a rule is not setup for the protected application in PingAccess, the policy is evaluated against the default Authentication adapter configured in this section. If a default adapter is not configured and a rule is not setup in PingAccess, authentication fails because an adapter is not configured for user authentication.
3. From the **Environment** drop-down list box, select an environment.
   The adapters for the selected environment load. Enter the adapter information for each environment that will be used by the adapter (For example, DEV and PROD)
4. Select an adapter from the drop down list box.
   The configuration syntax for the ACR selected displays.
5. Select a configuration to populate the configuration syntax button using PAPM.

   Note: HTML configuration is recommended for testing.

    - **Basic Configuration:** Sends the Authorization Header instead of a username and password in the request.
    - **HTML Configuration:** Sends username and password. The HTML configuration is recommended when the user is challenged for credentials with a basic pop-up.
6. Click **Save Configuration**.

Logic testing

Policy Test Tool does not support strong authentication. These adapters are pre-configured by PAPM but may require customizations by the user.

- Authentication Context (ACR)
- Default Authentication Adapter

PTT is capable of testing agent based and proxy based applications. See *Architecture overview* on page 417 to understand the difference between the two.

- **Select Test Configuration**: Select the environment, application, context, and test resource for the test. For site applications, the test resources should be a valid endpoint.
- **Evaluate Policy:** Use this section for testing policy logic using a single user. Provide the test user credentials. Check Groovy script option for Proxy based applications. See *Display headers for a proxy based application* on page 420 for more information. Click **Save** to evaluate the policy and display results.

Processing steps

After a test executes, a summary of the test results display on the right side of the screen. Click **View Processing Steps** to see a breakdown of the steps the tool executed. This is helpful to diagnose a test that does not complete. The step that caused the error will display at the bottom of the step list.

View test results

The **Test Results** tab contains an audit log of all Logic tests executed in the PTT. Load test do not display on this tab, they display on the *Load Test Results* tab.

- **HTML Return Content:** Click to view the HTML Return Content**.** The **Document View** can display download page content with a proxy format but not with the agent format. The **Raw Message** displays unformatted data returned.
- **User**: Displays the user ID that is sent to the application to execute the test.
- **Executor**: The user ID that logged into PAPM who configured and executed the test.

Execute a PingAccess logic test

About this task

For Proxy based applications, PingAccess acts as a gateway. Only the authorized requests reach the target web application server.

Agent based applications are intercepted at the target web application server by a PingAccess Agent. PingAccess Policy Server then makes the access decisions based on the policies configured.

Below are the steps to test Agent based or Proxy based policies using PTT.

Steps

1. Click the **PingAccess Test** link in the PTT.
2. In the **Select Test Configuration** section, select an environment and click the **Load** button.
   The applications for the selected environment populate the **Application** drop-down list box.
3. Select an application from the list.

   The list of resources associated with the selected application displays.
4. Select a resource.

   Note: The **lock** Icon indicates a protected context resource and the **unlock** icon indicates an unprotected context resource.

   The selected resource populates the **Test Resource** field.
5. If needed, change the **Test Resource** field to contain URL of the application to be tested**.**
6. Optional step: To add SSO headers in the response, check **Add Groovy Script** .
7. Click **Save**.

   Test results display on the right side of the screen.

   - If the user is successfully authenticated and authorized, the resource is rendered and the SSO headers are returned if Add Groovy Script is checked.
   - If the user is not authorized to access the resource, the test fails and generates a Forbidden error message.
   - In case the test fails, headers are not returned and an error message displays on the screen.

Execute a PingFederate logic test

Steps

1. Click the **PingFederate Test** link in the PTT.
   The **PingFederate Policy Test** screen displays.
2. In the **Select IDP Configuration** section, select an environment and click the **Load** button.
   The applications for the selected environment populate the **IDP Connection** drop-down list box.

3. Select an **IDP Connection** from the list, click the load button.
   The **Connection URL** and **Adapters** for the selection populates.
4. Select a resource.
   The selected resource populates the **Test Resource** field.
5. In the **Evaluate Policy** section, enter a username and password.
6. Click **Save**.
   Test results display on the right side of the screen. If successful, a green success banner will be displayed. Click on **SAMLResponse** to display decoded SAML Response.

Display headers for a proxy based application

When accessing a proxy-based protected resource, the SSO headers are returned to the protected application by checking the **Add Groovy script** option. The SSO Headers are included in the response object of PingAccess and display in the PTT.

It all happens behind the scenes. When the **Add Groovy Script** option is selected, the PTT automatically creates a new rule (PATESTOOL_SITEGROOVYSCRIPT) in PingAccess. This rule links to the protected application by creating a new policy for the protected resource. After the test is complete, the newly created policy deletes automatically. While the new rule still exists, it can be safely deleted.

Load testing

The multi-threaded load test benchmarks PingAccess and PingFederate policy execution and response times. Metrics are stored in a local audit database and available via an interactive graphic console. User lists configure to automate the testing of both successful and unsuccessful policy logic to ensure that the wrong users are not given access.

Load test calculations

The Thread Count, Requests per Thread, and AZ count are essentially load multipliers. Multiple all three values to get the total number of requests that will go to PingAccess or PingFederate.

Consider the following values for a Load Test with 5 different Test users (Threads) as an example:

```
Thread Count=5, Requests per thread=4, AZ Count=3.
```

In this configuration, 5 threads run 4 requests each (5*4) which yields 20 login attempts. Each login attempt has 3 Authorizations so total Authorizations to be performed is 5*4*3=60.

Load Test Policy

Input test configuration and load test policy.

View load test results

Historical data found in the **Load Test Results** list can be used to compare the performance results before and after an upgrade and assist in troubleshooting. PTT saves a history of test results in a database that includes the response time and authorization time of the protected resource. The result file is stored as a CSV file at `<PAPM_HOME>/config/PolicyTestTool/download`.

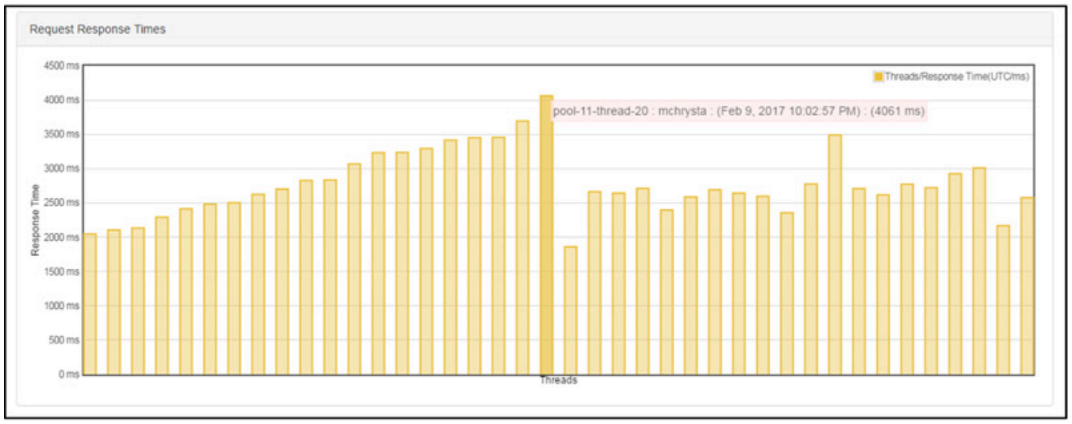
- Click the icon to view additional details of a load test.
- Click the **View Details** link to open the **Load Test Details** screen to view test results, charts, threads, and to download test results.

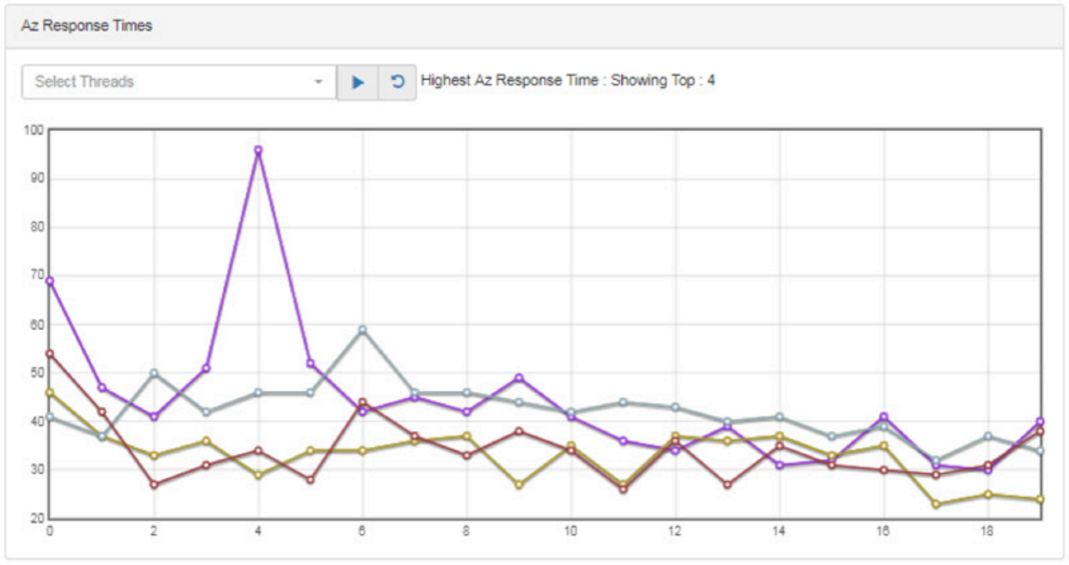
Load Test Details: Results

- **90% Percentile Az Requests**: Displays the value where the 90% of requests are below that value and only 10% are higher. For example, only 10% of the total Authorization Requests (200 in this case) took more than 461ms.
- **90% Percentile Response Time:** Displays the value where 90% of the response times are below that value and only 10% are higher. For example, only 10% of the response times were higher than 3457ms.

- **Average Az Time**: Average authorization time.
- **Average Response Time:** Average response time of the requests.
- **Failed Az Requests**: Total number of failed authorization requests.
- **Failed Requests**: Total number of failed requests. This includes Authentication failures or failures caused due to network issues.
- **Max Az Time**: Maximum time taken for authorization. Time taken by all the Authorizations is recorded and the one with the highest value is identified which gives the Max Authorization Time,1155 ms in this example.
- **Max Response Time:** Maximum response time taken by the request.
- **Successful Az Requests**: Total number of successful authorization requests.
- **Successful Requests**: Total number of successful requests.
- **Test Param : Application**: Name of the application being tested.
- **Test Param : Az Count**: Number of authorizations each thread will perform.
- **Test Param : EndTime**: Date and time when the load test was ended.
- **Test Param : Environment**: Environment for which the load test was performed.
- **Test Param : Executed By User**: Specifies the user who performed the load test. Displays in the Test Results list as Executor.
- **Test Param : Requests per Thread**: Specifies the number of requests each thread will make.
- **Test Param : Resource**: URL of the protected resource.
- **Test Param : StartTime**: Date and time when the load test was started.
- **Test Param : Status**: Status of the load test. For example, TERMINATED or COMPLETE. If the Load Test is TERMINATED, it specifies the username who terminated the test.
- **Test Param : Threads:** Total number of threads . In the above example, it is 20.

Load test details: Charts





Load Test Details: Threads

Displays information about the threads, number of users, used to execute the load test.

Load Test Details: Save Data

Click the **Save Data** link in the left navigation area on the **Load Test Details** screen to download a CSV file containing the Load Test Results for the selected test.

Execute a PingAccess load test

Steps

1. Follow the steps listed in *Execute a PingAccess logic test* on page 419
2. Click the **Load Test Policy** button.
3. Enter the Thread Count, Requests per thread and AZ Count as needed.
4. Add at least one user to the load test.
5. Click the **Load Test Policy**.
   The load test is executed. The status displays on the screen. If an error occurs, see *Managing error messages* on page 424 for more information.
6. Click **View Details**.
7. Click **Save Data** to save the results.
   The user is prompted to provide the location to save the results file in csv format.

Execute a PingFederate load test

About this task

Steps

1. Follow the steps listed in *Execute a PingFederate logic test* on page 419 to configure the policy to test.
2. Click the **Load Test Policy** button.
3. Enter the Thread Count and Requests per thread.
4. Add at least one user to the load test.
5. Click the **Load Test Policy**.
6. Click **View Details**.
7. Click **Save Data** on the Result screen to save the results.

   The user is prompted to provide the location to save the results file in csv format.

Add multiple users to a load test

About this task
An unlimited number of users can be added to a load test to simulate an event where multiple users attempt to access a policy at once. The users typed here are saved in a configuration file and will be available to use on every policy associated with the selected application. Each load test must have at least one test user.

Note: Use caution when loading test users because User Password information displays in plain text on the screen.

Steps

1. Using the Policy Test Tool, from the **Load Test Policy** screen, click the **Configure Load Test Users** button.
2. Enter a user name and password separated by a comma and press the **Enter** For example, `testuser.1,password`. User IDs and passwords are case sensitive.
   A carriage return is added to the end the line and the cursor moves to the next line.
3. To add another user, repeat step 2. The following example shows the credentials of three users: Test User, Jane.Doe, and Be.Hill.

   - `TestUser,password`
   - `Doe,secretepw`
   - `Hill,87dsljh`
4. Click **Save Users**.
5. Click the **Close (X**) icon to close the screen.

   The **Test Policy** screen displays.

Terminate a load test

About this task
Only one load test at a time is executed. To terminate a load test that is in process, follow the steps below.

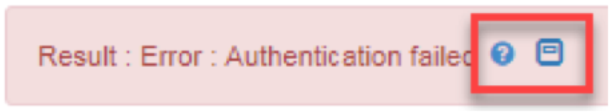
Steps

1. Using the Policy Test Tool, click **Test Policy** from the left navigation menu.

2. Expand **Test Policy**.
3. Click **Result**.
4. Click the **Load Test Results** tab.
5. Select the test and click **Terminate**.

Managing error messages

Error messages display in a light red box. Inside the message box, the Error Help Icon and HTML Return Content Icon contain additional information about the error.



Result : Error : Authentication failed

Policy Migration for CA SSO (Siteminder)

PAPM's PingAccess Policy Migration for CA SSO radically simplifies the migration process by automating policy transformations from CA Single Sign-On XPS policies to PingAccess and PingFederate. It creates working PingAccess policies from a SiteMinder policy export and integrates with the PingFederate and PingAccess APIs in a user-friendly manner. PAPM removes the guesswork from your policy migration and seamlessly transform CA SSO policies into to PingAccess.

Migration overview

SiteMinder XPSExport

CA SSO (SiteMinder) XPSExport is a tool to export the SiteMinder policy data that is included with the SiteMinder install. XPSExport creates an XML file which may have dependency information based on the exported policy. That's why it is important how this information is used for the target environment. When moving SiteMinder policies from one environment to another, either as part of an upgrade or a policy migration, some objects that are environment–specific are included in the export file. Examples of these objects include:

- Trusted hosts
- HCO Policy Server settings
- Authentication scheme URLs
- Password services redirects
- Redirect responses

See *Migration process* on page 425 for more information.

Configuring templates

The CA SSO policy migration is heavily dependent on the configured policy templates. CA SSO does not have a concept of OIDC for web applications or a web session in the same way that PingAccess and PingFederate use these attributes. As such, it is not possible to directly import or map these attributes. However, it is possible to setup an application template that can capture the requirements of your applications.

Remove imported XPS files

About this task
The following steps describe how to remove imported XPS files that are no longer needed.

Steps

1. Using Policy Migration for CA SSO, click **XPS Policy File** from the left navigation menu.

2. Click **Remove Imported Files.**
3. Click the **Delete** link for the file that you wish to delete.

Migration process

The following steps describe the migration process. See the *Troubleshooting* on page 440 section for troubleshooting tips.

- *Configure the XPS export file* on page 425
- *Create PingAccess template objects* on page 425
- *Create PingFederate template objects* on page 426
- *Create PAPM template group* on page 426
- *Upload the XPS export file* on page 426
- *Select the SiteMinder domain* on page 427
- *Map and import the policy domain* on page 427
- *Execute migration* on page 430
- *Validate migration* on page 430

Configure the XPS export file

About this task
A utility that is included with the SiteMinder install creates the XPS Export file.

Steps

Type the following command on the SiteMinder server to create a full back-up with all objects:

```
XPSExport.exe <filename> -xb
```

- To export a specific domain:

  ```
  XPSExport.exe <file name>-xo CA.SM::Domain@<xid>
  ```

- For example:

  ```
  XPSExport.exe D:\temp\policydata_domain.xml -xo CA.SM::Domain@03-8754fa84-
  c8ee-47a5-ba9d-14bec505fb02
  ```

Note: If the object xid is unknown, either lookup in the full back or use `XPSExplorer => option 79` for domain, the `S` for search, then copy the xid to the above command

Create PingAccess template objects

About this task
The templates in this step are created in PingAccess and are used for the migration in *Create PAPM template group* on page 426.

Tip: Use Policy Automation to create a PingAccess application policy. PAPM will auto generate all of the policy objects required. After the objects are created, you can modify them manually to meet your specific requirements.

Steps

1. Using PingAccess, create an agent object template per the policy requirements.
2. Using PingAccess, create a web session template object per the policy requirements.
3. Using PingAccess, create an identity mapping template per the policy requirements.

4. Using PingAccess, create a site object template per the policy requirements.

Create PingFederate template objects

About this task

Log into PingFederate to create OpenID Connect templates. These templates are referenced in *Create PAPM template group* on page 426.

Steps

1. Using PingFederate, create an OpenID Connect template object per the policy requirements.
2. Using PingFederate, create an OpenID Connect client object per the policy requirements.

Create PAPM template group

About this task

Set up template policy objects for all of the policy components. These templated components are grouped into a Policy Group in PAPM. This Policy Group is used to map the policy objects during the import phase.

Tip: Create a working application as a template to ensure that the LDAP lookups and objects are all able to work together.

There are two sections in the settings applet. The left pane creates new Policy group templates, and the right pane edits existing templates. The Policy Tool retrieves the template objects directly from your PingAccess and PingFederate admin servers.

Steps

1. Open Policy Migration for CA SSO.
   The SiteMinder XPS Policy File upload screen displays.
2. Using PAPM Policy Migration for CA SSO, click **Settings**.
3. From the **Environment** drop down list box, select the environment that contains the PingAccess and PingFederate templates for the migration create in *Create PingAccess template objects* on page 425 and *Create PingFederate template objects* on page 426.
   The template objects created for the environment load.
4. In the **Agent** box, select the agent object template for the migration created in *Create PingAccess template objects* on page 425.
5. In the **Web Session** box, select the web session object template for the migration created in *Create PingAccess template objects* on page 425.
6. In the **Identity Mapping** box, select the identity mapping template for the migration created in *Create PingAccess template objects* on page 425.
7. In the **Site** box, select the site object template for the migration created in *Create PingAccess template objects* on page 425.
8. In the **OIDC Policy** box, select the OpenID Connect policy template object for the migration created in *Create PingFederate template objects* on page 426.
9. In the **OIDC Client** box, select the OpenID Connect client template object for the migration created in *Create PingFederate template objects* on page 426.
10. Click **Create Policy Group**.

Upload the XPS export file

About this task

Steps

1. Using Policy Migration for CA SSO, click **XPS Policy File** from the left navigation menu.
   The **XPS File Upload** screen displays.
2. Click **Choose File** and select the XPS export file created in *Configure the XPS export file* on page 425.
   The file name displays in the **Select SiteMinder Policy File** box.
3. Click **Process File**.
   You are notified once the import is complete. If something is wrong with the file, a message indicating an invalid input file displays.
4. Click **Begin Migration** .

> **Note:** Processing times depend on the size of the SiteMinder domain configuration. If importing a small XPS file, then the policy migration can begin immediately. If importing an XPS file with over 500,000 lines, it may take up to two hours to process. Because this processing occurs in the background, the UI may incur a performance degradation until processing is complete.

Select the SiteMinder domain

About this task
Choose a SiteMinder domain to import in this step.

Steps

1. Using Policy Migration for CA SSO, go to the **Migrate Policy** screen and choose the domain file for the application in **Select Domain File**.
2. Select a Ping environment in the **Ping Environment** list.
   The domains contained in the XPS export file displays.
3. Select the SiteMinder domain you want to migrate by clicking the radio button in the first column.

> **Tip:** Navigate the list of domains by scrolling, using the page buttons below the list, or by searching for the domain you want to migrate. To search, click the Search dialog above the list to the right, and start typing. Searching is dynamic and your result set will narrow down as you enter more characters in the dialog.

4. Once you have selected the application domain you wish to migrate, click **Import Domain**.
   The domain is imported and the **Import Policy** screen displays

Map and import the policy domain

About this task
Once the export file has been uploaded and the policy domain is selected, map and import the policy domain on the Import Policy screen. The **Import Policy** UI provides the user with an interface to view the CA SSO policy objects and map them to a PingAccess and PingFederate application policy.

Steps

1. Using Policy Migration for CA SSO, complete the **Ping Environment** section on the **Import Policy** screen.

Field	Description
Ping Environment	Selected Ping environment. This value is based on the selected Ping Environment in the Settings Screen.

Ping Import Type	SiteMinder Migration allows two types, Site or Agent. Basing on the selected type, artifacts will change.
Ping Application Name	Application name imported from SiteMinder configuration file. Administrators can modify the Name as per the requirement.
Authentication Requirement	Authentication requirements are policies that dictate how an Administrator must authenticate before access is granted to a protected web resource. Create authentication requirements lists to identify these authentication methods. PAPM displays a list of one or more authentication methods which can be used across multiple application resources or it can be overridden by selecting individual authentication requirement in the application resources section. The values entered here must match the result values defined for the Requested AuthN Context Selector configured within PingFederate, and are ordered from highest to lowest value.
Policy Group	Select a Policy Group (Application Template) that best matches the user directory (or directories) tied to the CA SSO policy.
Agent Configuration	Click to display the SiteMinder Domain related Agent Config Objects (ACO).

2. From the **Application Resources** section on the **Import Policy** screen, select the protected / unprotected SiteMinder realms to import into PingAccess.

Field	**Description**
Select All	Selects all the application resources that will get created in PingAccess environment along with Application Creation. **Tip:** If the SiteMinder Policy Domain that you are importing includes realms (URL resources) from multiple applications, use the **Select All** checkbox to deselect all realms. Then use the **Search** box to search for the realms that belong in the application definition. Use the **Select All** checkbox to re-select the filtered realms.
Create	Creates the necessary application resource artifacts in the PingAccess Environment.
Resource Name	The resource name will be created in the PingAccess Environment. Administrators can customize the name. All text fields on this screen are editable.
Resource Path	Resources represent part of Web applications or APIs that have distinct security requirements. Resources specify what path prefixes and HTTP

	methods they apply to. Each application has at least the Root Resource which covers any requests not covered by other defined resources, and can have any number of additional resources. Resources may specify particular authentication requirements, including no authentication if the Anonymous checkbox is selected. Use this page to view and edit existing application resources and to define new resources.
Rules	Displays the rules if they are configured in SiteMinder Web Session Attributes. These Web Session attributes will be converted as Rules in PingAccess Application. Rules for access control and request processing.
Search	Search for a specific Application Resource. You can use search when there are multiple Application Resources and you want to search for a specific resource.
ACO- Ignore Extensions	Text box to enter any extensions to be ignored during the SiteMinder domain application migration. A resource is created called IgnoreExtensions that matches the values configured here.
Bad URL Chars & XSS Check	Filter for characters or CSS that may form a valid, executable script when displayed by the browser. A rule is created that blocks these characters.
Web Session Timeout	Contains the maximum time out and idle time out in minutes. PAPM populates this value with the lowest value found in selected application resources (SiteMinder Realms).

3. Complete the **Application Virtual Host/Agents** section on the **Import Policy** screen.

Field	**Description**
Virtual Host	These are the virtual host names and ports from which PingAccess accepts requests (e.g., myhost.com:80). A wildcard (*) is used to accept any host with the specified port (e.g., *:3000).
Site	These are site’s target application Host and Port number.
Identity Mapping	These mappings show the HTTP response headers that will be generated to the protected application. **Note:** If the Identity Mapping values do not match the values in the policy template or of the rules for the SiteMinder protected realms, then an error will be generated at import time. The policy will require some manual fixing to ensure that the attribute lookups match both the policy rules and the output HTTP headers.

Create Application	Create an application in the selected Ping Access Environment by using the pre-configured templates.

Execute migration

Steps

1. Using Policy Migration for CA SSO, click **Create Application**.
2. The results display on the **Results** screen.
3. From the **Results** page, open any item to view the REST API request and response created.

   This is useful in troubleshooting when an operation reports an error.

Validate migration

To validate the migration, access the migrated application.

Policy Migration for Oracle Access Manager

PAPM for Oracle Access Manager (OAM) optimizes the process of taking your existing Oracle Access Manager authentication and authorization policies, and migrating them to PingAccess and PingFederate. The tool simplifies the migration of OAM policies to PingAccess and PingFederate by exporting the policy domains from both versions to a common JSON format that PAPM can understand. This process helps remove the guesswork of how your application’s authentication and authorization policies map to the PingAccess and PingFederate format.

Note: The migration tool supports OAM version 10g and 11gR2, and has been specifically tested with OAM versions: 10.1.4.3 and 11.1.2.3. OAM 11g R1 has a different policy structure and is not supported.

Migration overview

Migrating OAM policies to Ping is a multi-step process. The setup of the tool and supporting components is a one-time activity, and the migration of policies is performed application by application via the user interface.

Migration Components

- **Export OAM Policy Store:** The OAM Policy Export Tool is a standalone Java program that can connect to either OAM 10g or 11g environments. Use the tool to export the OAM Policy Store to the PAPM JSON format. The JSON file is imported to the PAPM Migration Tool.
- **Load OAM Rule Processor Plugin:** The OAM Rule Processor Plugin for PingAccess adds support for the wildcard expressions used by OAM when defining URI resources for authorization. The OAM Policy Migration tool requires that this plugin be activated on all PingAccess servers before OAM policy migration takes place.
- **Configure Templates** is an important step in the Migration process as it directly reflects the PingAccess or PingFederate configurations which will be used by the migrated OAM policy. Administrators should verify the current OAM Domain requirements and replicate those configurations in PingAccess and PingFederate environments. Other settings that cannot be imported from OAM (because PingAccess and OAM work very differently) will be inherited directly from the template objects. The PAPM will use these configurations as a template to create the applications.
- **Import File and Select Domain**: After importing the JSON file from the OAM Policy Export Tool, an administrator will use the Select Domain Page to choose the application policy to be migrated. Selecting an OAM Domain and choosing to import will bring you to the Create Application Page.

- The **Select OAM Application** page shows the available Resources, associated Policies, and OAM Host Identifiers as they will be imported into the Ping Environment. This step will use the templates defined in Step 2 and create the applications in PingAccess and PingFederate environment.
- **Test**: After processing the Create Application Function, successfully created applications can be tested using Policy Test Tool to verify the migrated application policy.

Configuring Environments

PAPM works with existing PingAccess and PingFederate environments from development to QA or production. OAM Migration requires at least one pre-configured PingAccess and PingFederate environment before setting up migration templates described below. See *Settings* on page 396 for details on how to configure connectivity to your PingAccess and PingFederate servers.

Configuring Templates

Templates containing policy requirements are a critical component of the PingAccess Policy Migration for OAM tool. They contain PingAccess and PingFederate environment configurations for each type of application object.

As these objects are used during the application migration, you should check the current OAM domain configurations and create the equivalent artifacts in PingAccess as a template. The tool allows users to use templates multiple times for OAM Domains that share similar configurations.

OAM Policy Export Tool

The OAM Policy Export Tool supports both OAM 10g (10.1.4.3) and 11gR2 (11.1.2.3) environments.

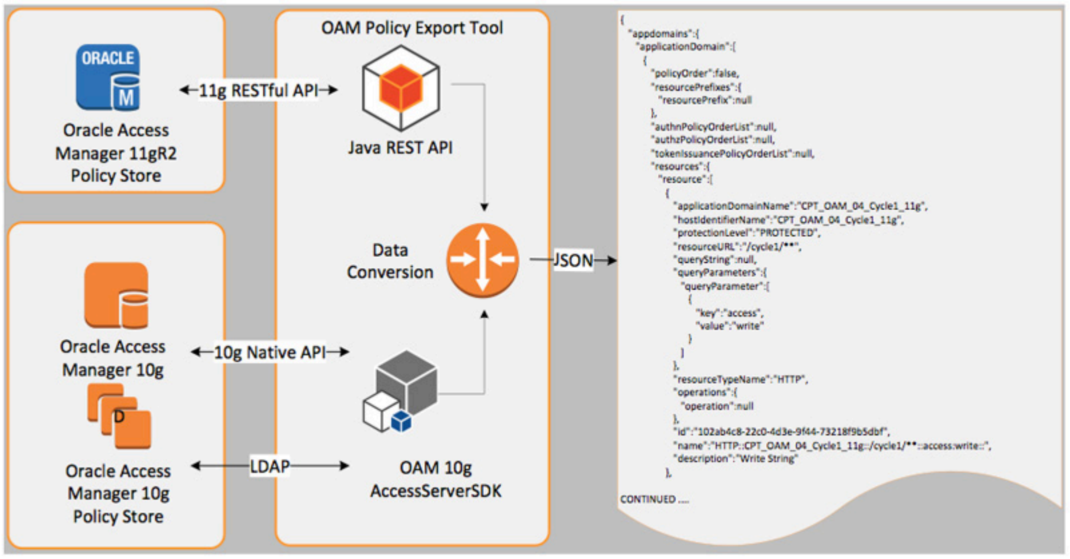
Note: The OAM Policy Domain Export Tool is a standalone tool independent of PAPM and is not accessed using the PAPM UI. The tool is located in the following location: `<PAPM_HOME>/extras/OAMExportUtility/`

The tool exports the following into a JSON formatted file:

- OAM Host Identifiers
- Policy Domains
- Defined Resources
- Authorization Policies
- Policy Responses

The OAM servers must be operational for the export tool to work. Each version uses specific Policy API calls to extract policy store data before storing the data in the export file.

Policy Domain Export Process



```
{
  "appdomains":{
    "applicationDomain":[
      {
        "policyOrder":false,
        "resourcePrefixes":{
          "resourcePrefix":null
        },
        "authnPolicyOrderList":null,
        "authzPolicyOrderList":null,
        "tokenIssuancePolicyOrderList":null,
        "resources":{
          "resource":[
            {
              "applicationDomainName":"CPT_OAM_04_Cycle1_11g",
              "hostIdentifierName":"CPT_OAM_04_Cycle1_11g",
              "protectionLevel":"PROTECTED",
              "resourceURL":"/cycle1/**",
              "queryString":null,
              "queryParameters":{
                "queryParameter":[
                  {
                    "key":"access",
                    "value":"write"
                  }
                ]
              },
              "resourceTypeName":"HTTP",
              "operations":{
                "operation":null
              },
              "id":"102ab4c8-22c0-4d3e-9f44-73218f9b5dbf",
              "name":"HTTP::CPT_OAM_04_Cycle1_11g::/cycle1/**::access:write::",
              "description":"Write String"
            },

CONTINUED ....
```

The OAM Policy Export Tool communicates with the **OAM 11gR2** environment via the REST Policy Administration API interface. To extract the OAM **10g policies**, the tool uses a combination of LDAP and Native OAM API calls via the AccessServerSDK.

OAM 11gR2 and **OAM 10g** follow a similar logical flow when exporting the data.

1. **Identify Application Domains**: Query the list of application domains
2. **Extract Data Cycles**: Iterate through each policy domain
   a. Extract all the hosts from the Host Identifiers
   b. Find all the defined resources
   c. Determine which resources are protected and which have anonymous access
   d. Extract all the authorization policies with allow and deny rules
   e. Extract all of the policy responses
3. **Format and Normalize Data**: Transform the version specific format to a common JSON format.
4. **Create JSON File**: Write the JSON formatted data to an output file.

The resulting JSON file contains the source data used by the PingAccess Policy Migration for OAM tool.

PAPM OAM Rule Plugin for PingAccess

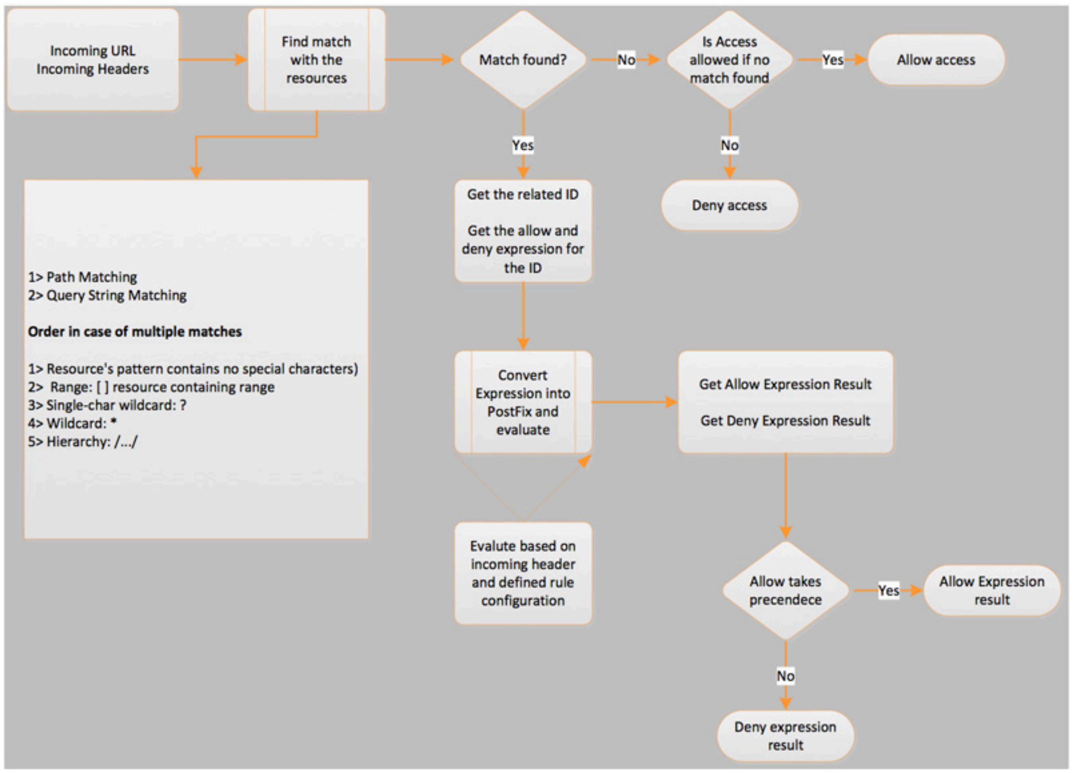
PingAccess and OAM do not support the same resource definitions. To bridge this gap, the PAPM OAM Rule Plugin for Ping Access was developed. It can handle advanced resource and policy definitions that use multiple wildcards and/or regex expressions to define protected or unprotected resources.

This plugin is configured on PingAccess servers prior to starting the application migration process so that the correct rules and policies are configured when migrating each application to Ping.

If an OAM policy contains authorization expressions, it will probably contain rules that use the PAPM OAM Rule Plugin. This plugin will evaluate URI resources with OAM type resource wildcards and query strings that Ping does not support out of the box.

Resource Matching Process

The PAPM Rule Processor follows the same logic for resource matching that Oracle Access Manager uses. This diagram shows the matching process for resources defined as a custom PingAccess rule using this plugin.


Incoming URL
Incoming Headers
Find match with the resources
Match found?
No
Is Access allowed if no match found
Yes
Allow access
Yes
No
Get the related ID
Get the allow and deny expression for the ID
Deny access
1> Path Matching
2> Query String Matching
Order in case of multiple matches
1> Resource's pattern contains no special characters)
2> Range: [ ] resource containing range
3> Single-char wildcard: ?
4> Wildcard: *
5> Hierarchy: /…/
Convert Expression into PostFix and evaluate
Get Allow Expression Result
Get Deny Expression Result
Evalute based on incoming header and defined rule configuration
Allow takes precendece
Yes
Allow Expression result
No
Deny expression result

Migration process

The following steps describe the migration process. See the *Policy Migration Tool for OAM* on page 442 section for troubleshooting tips.

- *Configure the OAM Policy Export Tool* on page 433
- *Export the OAM policy store* on page 434
- *Set up the OAM Rule Processor Plugin for PingAccess* on page 434
- *Create PingAccess template objects* on page 434
- *Create PingFederate template objects* on page 435
- *Create template object group* on page 435
- *OAM Policy Store file upload* on page 436
- *Select OAM application* on page 436
- *Configure import policy* on page 436
- *Execute migration* on page 438
- *Validate migration* on page 439

Configure the OAM Policy Export Tool

About this task

The OAM Policy Store export tool exports the entire policy store from OAM to a JSON formatted file. This file is then imported into the PingAccess Policy Migration tool for OAM, which is the next step of the process. The OAM Policy Export tool requires configuration to connect to an OAM environment to export the data. Please follow the steps below to configure the tool. For more information, see *OAM Policy Export Tool*.

Steps

1. Check that Java 1.8 is setup on the export utility server.

2. Copy the export tool from the following location to any server that has connectivity to your OAM Servers. The export tool is located in the following location:

```
<PAPM_HOME>/extras/OAMExportUtility/
```

> **Note:** For an **OAM 10g** export, AccessServerSDK is required. This is not required for OAM 11g exports. The SDK can be downloaded from Oracle at the following location: *https://docs.oracle.com/cd/E12530_01/oam.1014/e10355/as_api.htm*. An environment variable ACCESS_SERVER_SDK_HOME must be set to the AccessServerSDK installation path.

3. Update the OAM environment details in `application.properties` file that is located in the config folder for the OAMExportUtility (`<PAPM_HOME>/extras/OAMExportUtility/config/`).

> **Note:** Passwords saved as plain text are encrypted by the utility during the export operation.

Export the OAM policy store

About this task
Ensure that your OAM Export Tool has been setup according to the instructions in the previous section of this document.

Steps

1. To execute the export utility, navigate to the directory where you have setup your OAM Export Tool.
2. Run the export utility using <PAPM_HOME>/extras/OAMExportUtility/bin/oamexport.sh or .bat. Required parameters are as follows:

   OAM10g - `oam10g [OAM10GADMINPASSWORD] [OAM10GLDAPPASSWORD]`

   OAM11G - `oam11g [OAM11GPASSWORD]`

   Both - `both [OAM10GADMINPASSWORD] [OAM10GLDAPPASSWORD] [OAM11GPASSWORD]`

> **Note:** The exported data is stored in the `<PAPM_HOME>/extras/OAMExportUtility/output` folder as a JSON file, and will be named using the date and timestamp when the utility was executed.

> **Note:** Java warnings should not be a cause for concern.

Set up the OAM Rule Processor Plugin for PingAccess

About this task
Enable the PAPM custom plugin on all PingAccess servers in the environment prior to the migration. This ensures the custom authorization rules can be correctly instantiated on the PingAccess servers.

Steps

1. To configure the plugin , copy `<PAPM_HOME>/extras/OAMPARulePlugin/papm-oam-pa-az-rule-VERSION.jar` to the **lib** folder of ALL your PingAccess servers.
2. Restart each PingAccess server to enable the plugin.

Create PingAccess template objects

About this task
These templates are created in PingAccess and are used for the migration in Step 5.

Steps

1. Using PingAccess, create an agent object template using PingAccess per the policy requirements.
2. Using PingAccess, create a web session template object using PingAccess per the policy requirements.
3. Using PingAccess, create an Identity Mapping Template using PingAccess per the policy requirements.

> **Note:** OAM supports multiple types of policy responses including HTTP Headers, HTTP Cookies, and HTTP Session variables. PingAccess will only support HTTP headers or JSON web tokens. You will need to make sure your applications support the PingAccess model for identity mapping during the migration process.

4. Using PingAccess, create a site object template using PingAccess per the policy requirements.

Create PingFederate template objects

About this task
Log into PingFederate to create OpenID Connect templates. These templates are used in the next step.

Steps

1. Using PingFederate, create an OpenID Connect template object using PingAccess per the policy requirements.
2. Using PingFederate, create an OpenID Connect client object using PingAccess per the policy requirements.

> **Note:** Agents and Sites: PingAccess supports both Agent (Similar to an OAM Webgate) and Site (Reverse Proxy) based architectures. The Policy Group requires that you have a template for both types, and when you are in the process of migrating an OAM policy to Ping, you will have the ability to choose one or the other for applications as they are migrated.

Create template object group

About this task
Configuring templates is critical to a successful migration of OAM Policy Domains. The objects created in the PingAccess and PingFederate environments must meet the application requirements for user authentication, user attribute lookups, and session variables such as timeouts, scope, and HTTP headers. A template group is created in the PAPM that groups these template objects for the migration. The tool retrieves template objects directly from PingAccess and PingFederate admin servers.

Steps

1. Using the PAPM OAM Migration Tool, click **Settings**.
2. From the **Environment** drop down list box, select the environment that contains the PingAccess and PingFederate templates for the migration.
3. In the **Agent** box, select the agent object template for the migration.
4. In the **Web Session** box, select the web session object template for the migration.
5. In the **Identity Mapping** box, select the identity mapping template for the migration.
6. In the **Site** box, select the site object template for the migration.
7. In the **OIDC Policy** box, select the OpenID Connect policy template object.
8. In the **OIDC Client** box, select the OpenID Connect client template object for the migration.

9. Click **Create Policy Group**.

> **Note:** The Create New Policy group is used to create new policy group templates. The Edit Policy group is used for editing existing templates.

OAM Policy Store file upload

About this task
This step imports the JSON file created by executing the OAM Policy Export Tool in *Configure the OAM Policy Export Tool* on page 433.

Steps

1. Using the PAPM OAM Migration Tool, click **JSON File Upload**.
2. Click **Choose File** and select the JSON export file. The export file was named using the date and time that the file was created.
   The file name displays in the **Select OAm Policy File** box.
3. Click **Process**.

   Once the import is complete, you will be notified. If something is wrong with the file, a message indicating an invalid input file displays.
4. Click **Begin Migration** to proceed to the next step.

Select OAM application

About this task
Select the OAM application to migrate in this step. This page displays by clicking the Begin Migration button from the **JSON Policy File Upload** screen.

Steps

1. Using the PAPM OAM Migration Tool, from the **Select OAM application** screen, choose the domain file for the application in the **Select Domain File**
2. Select a Ping environment in the **Ping Environment** list.
3. Click **Display Domain**.
   The domains contained in the JSON export file display.
4. Select the OAM application domain that you wish to migrate by clicking the radio button in the first column. Navigate the list by scrolling, using the page buttons below the list, or by searching for the domain you want to migrate. To search, click the **Search** dialog above the list to the right, and start typing. Searching is dynamic and your result set will narrow down as you enter more characters in the dialog.
5. Once you have selected the application domain you wish to migrate, click the **Import Domain** button at the bottom of the screen.
   The domain is imported and the **Import Policy** screen displays.

Configure import policy

About this task
The **Import Policy** screen has several sections. Most of the data has been pre-populated from the OAM policy export. Choose the type of policy to create in the Ping environment, which template to use, and authentication requirements for the application resources. In addition, you can make modifications to the resources, environment details and identity mappings that are created in PingAccess.

Steps

1. Using the PAPM OAM Migration Tool, configure the Ping environment in the **Ping Environment** list.

Ping Environment	This setting is static and was set when you choose the OAM Domain to import on the previous screen.
Ping Import Type	Select the type of PingAccess application, Site or Agent. Data in the bottom section will change based on which type you choose here.
Policy Group	The Policy Group selection is populated with the templates you created. These templates will determine the policies created when the application is migrated to PingAccess and PingFederate.
Ping Application Name	The name of the OAM Policy Domain is populated in this field. Edit the name if needed.
Authentication Requirement	Select how PingFederate will authenticate users for the application.

2. Configure application URIs in the **Application Resources** screen. Specify whether they are accessible without authentication, or have associated allow and/or deny policies,

Create Checkbox	The first column contains a checkbox that indicates whether the resource and associated policy will be created in the Ping environment. Most resources will be selected by default with the following exceptions since they are specific to an OAM environment: ▪ Any resource beginning with /access/oblix ▪ Any resource beginning with /identity/oblix
Name	Each resource and policy is assigned a name. The associated authentication scheme is also listed for informational purposes. The scheme is not imported to Ping.
Path	This field contains the URI that will be imported to Ping. Modify the URL as needed before the application is created in the Ping environment.
Anonymous	Anonymous checkbox is available for those resources that have an anonymous authentication scheme assigned in your OAM policy domain.
Rules	The rules displayed in this section are based on configurations in the OAM policy domain. If specific users or groups (or LDAP filter rules) are assigned to allow or deny authorization rules in OAM, the corresponding rule will be imported to Ping.

3. Configure Virtual hosts, Sites/Agents, & Identity Mappings. The information in this section is populated by the OAM host identifiers and OAM policy responses.

Virtual Host	Contains the hostnames and port numbers that end users will use in URLs to access the application in PingAccess. The list is pre-populated with the complete list from the OAM Host Identifiers in the OAM Policy Domain.
Site / Agent	The middle portion of this section will change from site to agent based on the import yype chosen in the Ping Environment section at the top of the screen. The values populated are pulled from the site and agent templates defined on your settings screen described in Step 1. ▪ **Site:**– When the application is set as a site, then the application is protected using the PingAccess server as a reverse proxy, the host:port combinations listed here are the HTTP servers that the PingAccess server forward authenticated requests. ▪ **Agent: -** When the application is set as an agent, the application HTTP server is configured with a PingAccess Agent. The value listed here is pulled from the Agent Template and contains the host:port of the Agent listener on the PingAccess Server. ▪ **Identity Mapping:** Contains headers that are generated by PingAccess and injected into the HTTP response. The values listed here come from the OAM Policy Responses.

Note: OAM supports both HTTP Headers and HTTP Cookies while PingAccess supports HTTP Headers and Signed JWT Tokens. The migration tool supports the migration of HTTP Headers. It is possible that you will need to change this list to match your requirements, and developers of the application will have to change the application to support variables that PingAccess will support.

Ensure that your LDAP attributes that are listed here are valid in your LDAP identity store which is used by PingFederate. If PingFederate cannot match the attribute the migration will contain an error on the next step.

PingFederate is case sensitive when comparing attributes listed in this section to the LDAP Schema in the LDAP identity store. For example, if your list has an ldap attribute represented as ‘givename’ but your LDAP schema has ‘giveName’, then your migration will have an error.

Execute migration

Steps

1. Using the PAPM OAM Migration Tool, review the settings and click the **Create Application** button at the bottom of the page to start the migration.

The PAPM OAM Migration Tool will create all the supporting objects within PingFederate and PingAccess.

Migration results display on the results page.

2. From the **Results** page, open any item to view the REST API request and response created.

   This is useful in troubleshooting when an operation reports an error.

Validate migration

To test the migrated policy, access the migrated application.

Disaster recovery

High availability is the measurement of a system's ability to remain accessible in the event of a policy tool failure. By design, PAPM can only run on a single server. In the event of system failure, it is possible to copy the PAPM `config` folder to another server and bring up the PAPM.

The purpose of this section is to describe the configuration of the PAPM for high availability and disaster recovery by providing an understanding of the following:

- **Communication:** The PAPM communicates with PingAccess and PingFederate using standard APIs.
- **Data storage:** The PAPM stores the data in the *`PAPM_HOME`*`/data` directory and settings are stored in various properties files in the `config` file.
- **Back up process:** PAPM data is archived daily at approximately midnight server time to the following directory: `<PAPM_HOME>/data/backup`

What to Save

All files that need to be saved are located under the `<PAPM_HOME>/config` and `<PAPM_HOME>/data` directories.

Note: PAPM is typically recoverable if these directories are backed up.

Important: PAPM only retains administrative configuration prior to that configuration being pushed to PingAccess and PingFederate operational environments. As it is not required for runtime, the risk of PAPM being unavailable is restricted to loss of access to the configuration retained in PAPM. It does not affect either PingAccess or PingFederate operational availability or performance when unavailable.

Set up disaster recovery (data backup)

About this task

The following example provides steps that can be followed as a simple mechanism to provide an example of automated backup.

Steps

1. Setup a cronjob to archive the `<PAPM_HOME>/config` and `<PAPM_HOME>/data` folders daily or weekly.
2. Install the PAPM on a server.
3. SCP/FTP the `<PAPM_HOME>/config` and `<PAPM_HOME>/data` folders to the directory *`product path`* in the target server.
4. Unzip/UnTAR the archive file.
5. Start PAPM by going to *`<PAPM_HOME>/bin`* and invoke `run.bat/run.sh`.
6. Test PAPM console access via your browser and verify the configurations.

Troubleshooting

All of the tools in PAPM provide online troubleshooting functions. This section details the best ways to troubleshoot errors with each of the tools.

- *PAPM logging* on page 440
- *Login issues* on page 440
- *Policy Migration Tool for CA SSO* on page 442
- *Policy Migration Tool for OAM* on page 442

PAPM logging

The PAPM logger creates a series of organized log files for application troubleshooting.

Log Name	Description
`application.log`	The `application.log` file contains all of the standard Java log output from PAPM. Log Format: `DATE-TIME-LOG_LEVEL-CLASS:-LOG MESSAGE`
`application-api.log`	The `application-API.log` file contains the auditable requests to PAPM. Log Format: `DATE TIME|COMPONENT|ADMIN USER|API REQUEST URL|TARGET IP|METHOD (GET,POST)|RESPONSE CODE|RESPONSE TIME` Delimiter: Pipe

Login issues

Before you begin

If the LDAP account is misconfigured and access to the PAPM is locked out, you can manually reset the login back to the OOTB file-based login module.

Steps

1. Using Terminal, edit the properties file located at *`PAPM_HOME`*`/config/application.properties`.

   Set: `papm.force-database-authentication=true`
2. Restart PAPM.

Policy Automation

Policy Automation contains two sets of audit records:

- Successful releases (affecting live data and functions)
- Other audit records: Other API calls and **failed API transactions**

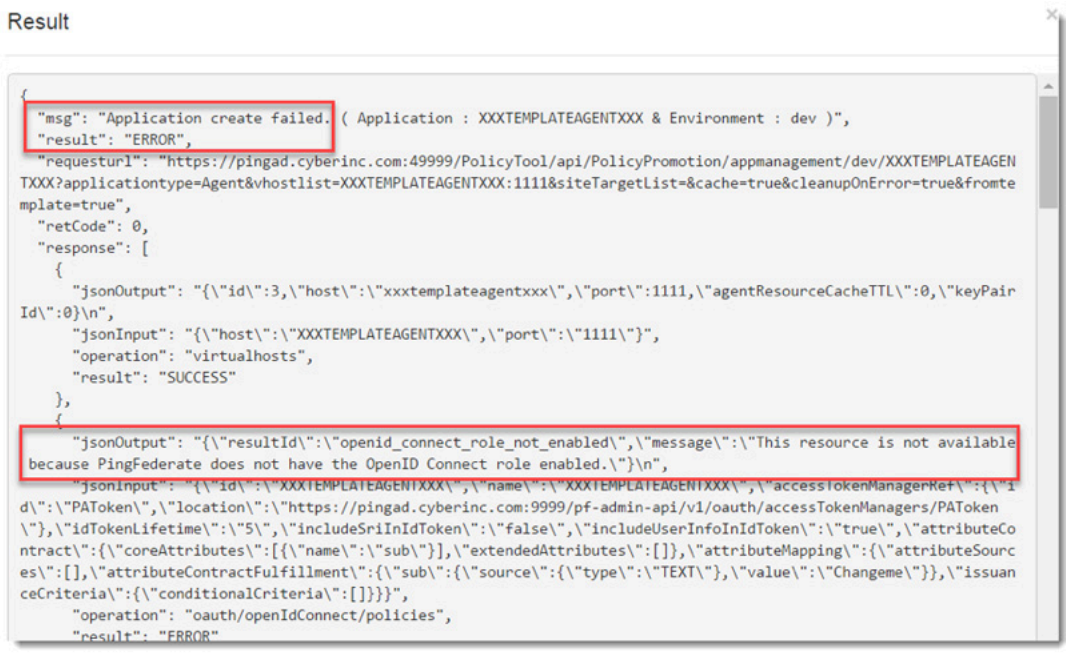
The **Other Audit** section is invaluable for troubleshooting errors in the Policy Promotion Tool.

1. **Using Policy Automation Tool > Audit Data** click **Other Audit** to open the Audit Data screen.

   Tip: It may be useful to search for FAILURE messages on this tab.

2. Click the link with the application name, for instance, XXXTEMPLATEAGENTXXX.

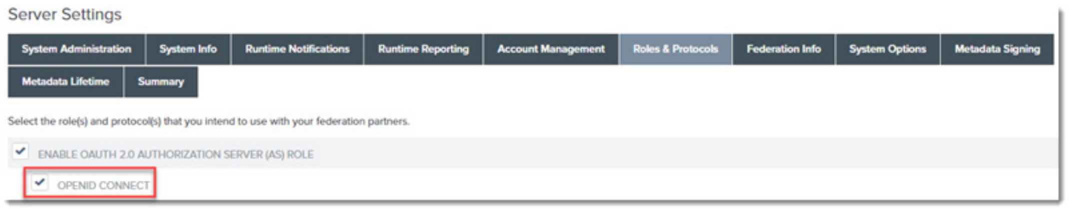
   This link displays the RESTful API request and response output. The output below contains a failed result.



Result

```
{
  "msg": "Application create failed. ( Application : XXXTEMPLATEAGENTXXX & Environment : dev )",
  "result": "ERROR",
  "requesturl": "https://pingad.cyberinc.com:49999/PolicyTool/api/PolicyPromotion/appmanagement/dev/XXXTEMPLATEAGEN
TXXX?applicationtype=Agent&vhostlist=XXXTEMPLATEAGENTXXX:1111&siteTargetList=&cache=true&cleanupOnError=true&fromte
mplate=true",
  "retCode": 0,
  "response": [
    {
      "jsonOutput": "{\"id\":3,\"host\":\"xxxtemplateagentxxx\",\"port\":1111,\"agentResourceCacheTTL\":0,\"keyPair
Id\":0}\n",
      "jsonInput": "{\"host\":\"XXXTEMPLATEAGENTXXX\",\"port\":\"1111\"}",
      "operation": "virtualhosts",
      "result": "SUCCESS"
    },
    {
      "jsonOutput": "{\"resultId\":\"openid_connect_role_not_enabled\",\"message\":\"This resource is not available
because PingFederate does not have the OpenID Connect role enabled.\"}\n",
      "jsonInput": "{\"id\":\"XXXTEMPLATEAGENTXXX\",\"name\":\"XXXTEMPLATEAGENTXXX\",\"accessTokenManagerRef\":{\"i
d\":\"PAToken\",\"location\":\"https://pingad.cyberinc.com:9999/pf-admin-api/v1/oauth/accessTokenManagers/PAToken
\"},\"idTokenLifetime\":\"5\",\"includeSriInIdToken\":\"false\",\"includeUserInfoInIdToken\":\"true\",\"attributeCo
ntract\":{\"coreAttributes\":[{\"name\":\"sub\"}],\"extendedAttributes\":[]},\"attributeMapping\":{\"attributeSourc
es\":[],\"attributeContractFulfillment\":{\"sub\":{\"source\":{\"type\":\"TEXT\"},\"value\":\"Changeme\"}},\"issuan
ceCriteria\":{\"conditionalCriteria\":[]}}}",
      "operation": "oauth/openIdConnect/policies",
      "result": "ERROR"
```

In this example, the OpenID connect role was not enabled on this server:



Select the role(s) and protocol(s) that you intend to use with your federation partners.

✓ ENABLE OAUTH 2.0 AUTHORIZATION SERVER (AS) ROLE

✓ OPENID CONNECT

This method helps troubleshoot many types of API errors, such as missing data, orphaned-linked records that cannot be deleted, incompatible data (across Ping API versions), etc.

Policy Test Tool

The Policy Test Tool provides an interface to:

- Test policy function
- Provide performance metrics
- Enable load testing of policy logic

Since the PTT is a testing tool, expect error conditions. It is important to recognize where the Policy Test Tool will show a success and when it will encounter issues. For example, strong authentication is not supported; resources with policy requirements for strong authentication will always fail.

Note: Strong authentication is not supported.

Ports, Connectivity

Unlike the other utilities that require API access, the Policy Test Tool additionally requires direct access to the proxy, login, and agent ports for the PingAccess and PingFederate servers for each environment. These ports are typically:

- PingFederate Engine: 9031
- PingAccess Proxy: 3000
- PingAccess Agent: 3030

Agent vs. Proxy Testing

The Policy Test Tool contains an embedded PingAccess Agent. This enables testing without ever deploying the application to the network. Any protected resource can be tested for authorization, response times, and mapped identity headers.

Agent Testing

Note the 'Agent' indicator in green. Any test resource should work, even resources with an '*' instead of a true URL endpoint.

Proxy Testing

Unlike the Agent, a Proxy-Based test resource **MUST** go to a real resource on the network and a true application page. It is not possible to virtually test proxy resources in the same way as an agent model due to the PingAccess system design.

The resource must be a true application endpoint or it will result in an error.

Policy Migration Tool for CA SSO

The Policy Migration Tool for CA SSO has a couple of features to troubleshoot policies online. However, it is important to ensure that the process begins with complete data.

Incomplete XPS export

The SiteMinder XPS Export should use the `-b` backup option to capture all data. If a partial policy backup is taken, some data may be missing. More importantly, the export data may be missing the Agent Configuration Object (ACO). This will effectively disable any policy import of unprotected resource extensions (ignoreExtensions) and the cross site scripting characters.

Poor performance:

If importing a large XPS Export file (>20MB), wait one to two hours before continuing to choose a domain. A time consuming background database process needs to parse hundreds of thousands of lines of XML and transform the data into a JSON based format. Once the process is complete, the tool should be very responsive.

Incorrect user store:

The migration process builds HTTP header and authorization lookups from a template policy. However, if the template policy does not link to the correct user directory or database, the migration process will build a stubbed OIDC policy (as it cannot find the appropriate data to map). Ensure that the template links to the same user store as the imported policy.

Missing HTTP response headers:

As mentioned above, the import process uses a template. If the template does not include the appropriate lookups (even to the correct user store) a valid mapping cannot occur. This will also result in stubbed attributes in the OIDC policy as well as attributes PingAccess identity mapping object that cannot be filled.

Note: Once a migration has completed, the results screen displays a list of the created objects. Each of these objects can be click to inspect the data objects used to create the policies. If you navigate away from the results screen, then simply go to the **Audit Data** menu under the SiteMinder Migration Tool.

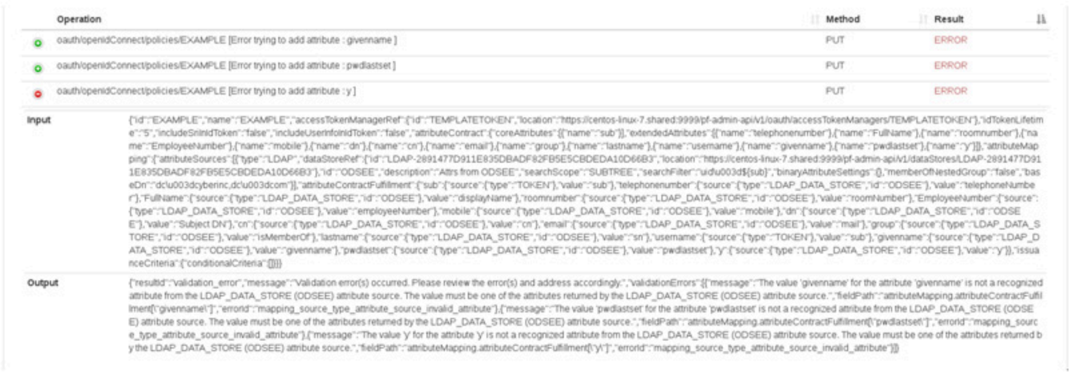
Policy Migration Tool for OAM

Below are some common issues that may be encountered when using the PingAccess Policy Migration tool for OAM.

Domain import failure

When an OAM application domain is imported into PingAccess, the import is not one operation, but many calls to the Ping REST APIs. If any of these API calls report an error, then the OAM domain migration will report a failure.

It is important to understand that when a failure is reported; it may be because an API returned an error response. Examine the results to find what has reported an error.



	Operation	Method	Result
●	oauth/openidConnect/policies/EXAMPLE [Error trying to add attribute : givenname]	PUT	ERROR
●	oauth/openidConnect/policies/EXAMPLE [Error trying to add attribute : pwdlastset]	PUT	ERROR
●	oauth/openidConnect/policies/EXAMPLE [Error trying to add attribute : y]	PUT	ERROR

Input

```
{"id":"EXAMPLE","name":"EXAMPLE","accessTokenManagerRef":{"id":"TEMPLATETOKEN","location":"https://centos-linux-7.shared:9999/pf-admin-api/v1/oauth/accessTokenManagers/TEMPLATETOKEN"},"idTokenLifetim
e":"5","includeSriIdToken":"false","includeUserInfoIdToken":"false","attributeContract":{"coreAttributes":[{"name":"sub"}],"extendedAttributes":[{"name":"telephonenumber"},{"name":"FullName"},{"name":"roomnumber"},{"na
me":"EmployeeNumber"},{"name":"mobile"},{"name":"dn"},{"name":"cn"},{"name":"email"},{"name":"group"},{"name":"lastname"},{"name":"username"},{"name":"givenname"},{"name":"pwdlastset"},{"name":"y"}]},"attributeMap
ping":{"attributeSources":[{"type":"LDAP","dataStoreRef":{"id":"LDAP-2891477D911E835DBADF82FB5E5CBDEDA10D66B3","location":"https://centos-linux-7.shared:9999/pf-admin-api/v1/dataStores/LDAP-2891477D91
1E835DBADF82FB5E5CBDEDA10D66B3"},"id":"ODSEE","description":"Attrs from ODSEE","searchScope":"SUBTREE","searchFilter":"uid\u003d${sub}","binaryAttributeSettings":{},"memberOfNestedGroup":"false","bas
eDn":"dc\u003dcyberinc,dc\u003dcom"}],"attributeContractFulfillment":{"sub":{"source":{"type":"TOKEN"},"value":"sub"},"telephonenumber":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"telephoneNumbe
r"},"FullName":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"displayName"},"roomnumber":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"roomNumber"},"EmployeeNumber":{"source":
{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"employeeNumber"},"mobile":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"mobile"},"dn":{"source":{"type":"LDAP_DATA_STORE","id":"ODSE
E"},"value":"Subject DN"},"cn":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"cn"},"email":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"mail"},"group":{"source":{"type":"LDAP_DATA_S
TORE","id":"ODSEE"},"value":"isMemberOf"},"lastname":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"sn"},"username":{"source":{"type":"TOKEN"},"value":"sub"},"givenname":{"source":{"type":"LDAP_D
ATA_STORE","id":"ODSEE"},"value":"givenname"},"pwdlastset":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"pwdlastset"},"y":{"source":{"type":"LDAP_DATA_STORE","id":"ODSEE"},"value":"y"}},"issua
nceCriteria":{"conditionalCriteria":[]}}}
```

Output

```
{"resultId":"validation_error","message":"Validation error(s) occurred. Please review the error(s) and address accordingly.","validationErrors":[{"message":"The value 'givenname' for the attribute 'givenname' is not a recognized
attribute from the LDAP_DATA_STORE (ODSEE) attribute source. The value must be one of the attributes returned by the LDAP_DATA_STORE (ODSEE) attribute source.","fieldPath":"attributeMapping.attributeContractFulfil
lment[\"givenname\"]","errorId":"mapping_source_type_attribute_source_invalid_attribute"},{"message":"The value 'pwdlastset' for the attribute 'pwdlastset' is not a recognized attribute from the LDAP_DATA_STORE (ODSE
E) attribute source. The value must be one of the attributes returned by the LDAP_DATA_STORE (ODSEE) attribute source.","fieldPath":"attributeMapping.attributeContractFulfillment[\"pwdlastset\"]","errorId":"mapping_sourc
e_type_attribute_source_invalid_attribute"},{"message":"The value 'y' for the attribute 'y' is not a recognized attribute from the LDAP_DATA_STORE (ODSEE) attribute source. The value must be one of the attributes returned b
y the LDAP_DATA_STORE (ODSEE) attribute source.","fieldPath":"attributeMapping.attributeContractFulfillment[\"y\"]","errorId":"mapping_source_type_attribute_source_invalid_attribute"}]}
```

The screenshot above shows a failed operation expand. This view shows the API input and the output from the PingAccess server in detail and why the request failed.

You can address these errors one by one directly in PingAccess and PingFederate environments or delete the application using the PAPM Policy Tool, and fix the errors during the migration process or policy templates.

Identity mapping errors

Mapping OAM policy responses to PingAccess identity mapping can be tricky. This operation can fail for multiple reasons, and proper planning and understanding of your application requirements will help avoid errors in this area.

Common reasons for errors creating PingAccess identity mapping policies are:

- **Case sensitive attributes**: While most directories are not case sensitive when interacting with LDAP schema, the PingFederate and PingAccess servers are case sensitive when setting attributes and header names.
  - **Example**: givenName NOT givenname
- **Attempting to map OAM cookie responses to PingAccess**: When the OAM Policy Import executes, all policy responses are categorized to the Identity Mapping section on the migration screen. If you have a cookie response assigned to an OAM policy, this will be created as an HTTP header in the PingAccess Identity Mapping.
- **Static and dynamic attributes:** Many times OAM policy responses pull the response value from a user's attributes dynamically. Other times there is a static value assigned to the response. Static values are not differentiated from dynamic attribute values in the migration tool, and must be accounted for when migrating OAM applications. Static values will be treated as LDAP attributes if they are not modified or removed completely.
  - **Example**: Cookie or header response set as LOGGEDIN=TRUE. The migration tool will send this request to PingAccess, and PingAccess will treat TRUE as an LDAP attribute, and therefore fail.
- **Mismatch of attribute names between OAM policy and PingAccess policy templates**: Attributes that are defined in the PingAccess templates need to match the name assigned in the schema. If these do not match, and you try to map an attribute that is not part of the LDAP lookup in the OIDC policy contract, then there is a possibility that the Identity Mapping for that attribute will fail.

Virtual host errors

Values that are populated in the virtual hosts section of the migration come from OAM host identifiers. Over time, the OAM host identifier list is often added to when servers migrate or are upgraded over time. However, it is common that older server hostnames or IP addresses are not removed.

When the PingAccess Policy Migration tool for OAM, pushes the list of hosts to virtual hosts in PingAccess, every DNS hostname is resolved to ensure it is a valid hostname. If an invalid hostname is found, then the operation will fail, and the Virtual Hosts will need to be created manually.

Site errors

When migrating an OAM application domain to a site (reverse proxy) based application in PingAccess, the list of hostnames in the site section of the migration tool is populated from the OAM host identifiers. This represents the application HTTP servers that the authenticated and authorized requests are forwarded to by the PingAccess proxy server.

This presents the same issue as described for virtual host errors above. When the PingAccess Policy Migration tool for OAM pushes the list of hosts to the site target in PingAccess, every DNS hostname is resolved to ensure it is a valid hostname. If an invalid hostname is detected, then the operation will fail, and the site targets will need to be created manually.